

User Guide GD2827

How to make coincidences with CAEN Digitizers

The complete guide with step-by-step instructions

Rev. 3 - March 14th, 2017

Purpose of this Guide

This guide wants to provide detailed instructions on how to implement on-line coincidences with CAEN digitizers and MCA equipped either with Default firmware for waveform recording or DPP (Digital Pulse Processing) firmware.

Change Document Record

Date	Revision	Changes
September 23 rd , 2013	00	Initial Release
July 4 th , 2016	01	Fully revised version. Added support to MC ² Analyzer. Added support to x730, x725, 781, and 790 series. DPP-CI no longer supported
November 22 nd , 2016	02	Modified register 0x1n84 for DPP-PSD of 725 and 730 series (revision 4.11_136.10 on). Modified register 0x1nA0 for DPP-PHA of 725 and 730 series (revision 4.11_139.5 on). Added example for 780 series. Modified bits[11:10] of register 0x811C. Corrected examples for 725-730 DPP-PHA. Fully revised Chapter 5 in "How to Veto the acquisition".
March 14 th , 2017	03	Modified examples for DPP-PSD of 725 and 730 series: latency window inside the couple set to 2 clock cycles; trigger validation mask corrected to 0xFFFFFFFF; added example for coincidences among 16 channels (VME).

Symbols, abbreviated terms and notation

ADC	Analog-to-Digital Converter
DAQ	Data Acquisition
DPP	Digital Pulse Processing
DPP-CI	DPP for Charge Integration
DPP-PHA	DPP for Pulse Height Analysis
DPP-PSD	DPP for Pulse Shape Discrimination
MCA	Multi-Channel Analyzer
OS	Operating System
PC	Personal Computer
PMT	Photo Multiplier Tube
QDC	Charge-to-Digital Converter
TDC	Time-to-Digital Converter
USB	Universal Serial Bus

Reference Documents

- [RD1] UM2088 – DPP-PSD User Manual.
- [RD2] UM3182 – DPP-PHA and MC² Analyzer User Manual.
- [RD3] W. R. Leo. Techniques for Nuclear and Particle Physics Experiments. Ed. by Springer. II ed.
- [RD4] K. S. Crane. Introductory nuclear physics. Ed. by J. Wiley and sons.
- [RD5] G. F. Knoll. Radiation detection and measurement. Ed. by J. Wiley and sons. III ed.
- [RD6] AN2086 – Synchronization of a multi-board acquisition systems with CAEN digitizers.
- [RD7] UM4855 – 720 DPP-PSD Registers Description.

- [RD8] UM5110 – 751 DPP-PSD Registers Description.
- [RD9] UM4380 – 725-730 DPP-PSD Registers Description.
- [RD10] UM5416 – DT5790 DPP-PSD Registers Description.
- [RD11] UM5678 – 725-730 DPP-PHA Registers Description.
- [RD12] CAEN Technical Information Manual. Mod. V1761.

All CAEN documents can be downloaded at: <http://www.caen.it/csite/LibrarySearch.jsp>

CAEN S.p.A.
Via Vetraia, 11 55049 Viareggio (LU) - ITALY
Tel. +39.0584.388.398 Fax +39.0584.388.959
info@caen.it
www.caen.it

©CAEN SpA – 2017

Disclaimer

No part of this manual may be reproduced in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of CAEN SpA.

The information contained herein has been carefully checked and is believed to be accurate; however, no responsibility is assumed for inaccuracies. CAEN SpA reserves the right to modify its products specifications without giving any notice; for up to date information please visit www.caen.it.

MADE IN ITALY: We stress the fact that all the boards are made in Italy because in this globalized world, where getting the lowest possible price for products sometimes translates into poor pay and working conditions for the people who make them, at least you know that who made your board was reasonably paid and worked in a safe environment. (this obviously applies only to the boards marked "MADE IN ITALY", we cannot attest to the manufacturing process of "third party" boards).



Index

Purpose of this Manual	2
Change document record	2
Symbols, abbreviated terms and notation	2
Reference Documents	3
1 Introduction	7
On-line coincidences: the traditional approach	8
2 Coincidences with CAEN DPP firmware	12
DPP Normal Acquisition mode	13
DPP Coincidence mode	15
Resolving time	18
Double coincidence	18
3 Instructions and Examples for DPP-PSD firmware	19
DPP-PSD Control Software	19
How to configure the registers - 720 and 751 series with DPP-PSD	20
Examples for 720-751 series	23
Coincidence between channel 0 and channel 1 (ch0 & ch1)	23
Coincidence among four channels (ch0 & ch1 & ch2 & ch3)	23
Coincidence between channel 0 and channel 1, coincidence between channel 2 and channel 3 (ch0 & ch1), (ch2 & ch3)	24
Coincidence among eight channels (ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7) - VME only	25
How to configure the registers - 725 and 730 series with DPP-PSD	26
Examples for 725-730 series	27
Coincidence between channel 0 and channel 1 (ch0 & ch1)	27
Coincidence inside the couples (ch0 & ch1), (ch2 & ch3), (ch4 & ch5), (ch6 & ch7)	28
Coincidence inside the couples (ch0 & ch1), (ch2 & ch3), (ch4 & ch5), (ch6 & ch7), (ch8 & ch9), (ch10 & ch11), (ch12 & ch13), (ch14 & ch15) - VME only	29
Coincidence among four channels (four different couples, ch0 & ch2 & ch4 & ch6)	30
Coincidence among four channels (two different couples, ch0 & ch1 & ch2 & ch3)	31
Coincidence among eight channels (eight different couples, ch0 & ch2 & ch4 & ch6 & ch8 & ch10 & ch12 & ch14)	32
Coincidence among eight channels (four different couples, ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7)	33
Coincidence among sixteen channels (ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7 & ch8 & ch9 & ch10 & ch11 & ch12 & ch13 & ch14 & ch15) - VME only	34
Majority of at least three couples among four couples (eight channels)	36
How to configure the registers - DT5790 with DPP-PSD	37
Examples for DT5790	37
Coincidence between channel 0 and channel 1 (ch0 & ch1)	37
Anti-coincidence between channel 0 and channel 1	38
4 Instructions and Examples for DPP-PHA firmware	39
MC ² Analyzer Software	39
Examples with MC ² Analyzer Software and 724-780-781 series with DPP-PHA	41
Coincidence between channel 0 and channel 1	41
Anti-coincidence between channel 0 and channel 1	42
Coincidence among four channels (from channel 0 to channel 3)	43

How to configure the registers - 724 and 781 series with DPP-PHA	44
Examples for 724 and 781 series with DPP-PHA	48
Coincidence between channel 0 and channel 1	48
Anti-coincidence between channel 0 and channel 1	49
Coincidence among four channels (ch0 & ch1 & ch2 & ch3)	50
How to configure the registers - 780 series with DPP-PHA	51
Examples for 780 series with DPP-PHA	51
Coincidence between channel 0 and channel 1	51
Anti-coincidence between channel 0 and channel 1	52
How to configure the registers - 725 and 730 series with DPP-PHA	53
Examples for 725-730 series with DPP-PHA	58
AND between channel 0 and channel 1 (ch0 & ch1)	58
Coincidence inside the couples (ch0 & ch1), (ch2 & ch3), (ch4 & ch5), (ch6 & ch7)	59
Coincidence inside the couples (ch0 & ch1), (ch2 & ch3), (ch4 & ch5), (ch6 & ch7), (ch8 & ch9), (ch10 & ch11), (ch12 & ch13), (ch14 & ch15) - VME only	60
Coincidence among four channels (four different couples, ch0 & ch2 & ch4 & ch6)	62
Coincidence among four channels (two different couples, ch0 & ch1 & ch2 & ch3)	63
Coincidence among eight channels (eight different couples, ch0 & ch2 & ch4 & ch6 & ch8 & ch10 & ch12 & ch14)	64
Coincidence among eight channels (four different couples, ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7)	66
Coincidence among sixteen channels (ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7 & ch8 & ch9 & ch10 & ch11 & ch12 & ch13 & ch14 & ch15) - VME only	67
Majority of at least three couples among four couples (eight channels)	69
5 How to Veto the acquisition	70
Signal on channel 0 vetoed by the other channels	70
Veto on ch0 - DPP-PSD for 720 and 751 series	71
Veto on ch0 - DPP-PSD for 725 and 730 series	72
Veto on ch0 - DPP-PHA for 724 and 781 series	73
Veto on ch0 - DPP-PHA for 725 and 730 series	73
Veto for the couples of channels (725-730 only)	74
Coincidence inside the VME couples (ch-i & ch-i+1) vetoed by the majority of two or more couples (DPP-PSD)	75
Coincidence inside the VME couples (ch-i & ch-i+1) vetoed by the majority of two or more couples (DPP-PHA)	77
External Trigger to veto (gate) the acquisition	79
Veto from External Trigger - 720 and 751 series with DPP-PSD	80
Veto from External Trigger - 725 and 730 series with DPP-PSD	81
Veto from External Trigger - 724, 780 and 781 series with DPP-PHA	81
Veto from External Trigger - 725 and 730 series with DPP-PHA	83
6 Coincidences with Default Firmware	84
7 Technical Support	86

List of Figures

Fig. 1.1 Analog coincidence scheme. The two pulses are summed and the discrimination level is set on the sum pulse, corresponding to the coincidence of the two	8
Fig. 1.2 The leading edge timing may suffer from amplitude walk problems	8

Fig. 1.3	The zero crossing timing of bipolar pulses	9
Fig. 1.4	The constant fraction timing method. From top to bottom the input pulse is split into an attenuated signal and a delayed and inverted one. The two are summed and the zero crossing time is taken as the time trigger	9
Fig. 1.5	Schematic example showing how to record coincidences of the same event from two different detectors	10
Fig. 1.6	Time spectrum for a radioactive source as in the configuration of Fig. 1.5	10
Fig. 1.7	The coincidence rate vs delay curve	11
Fig. 2.1	Example of digital chain for coincidences with CAEN digitizer	12
Fig. 2.2	DPP firmware functional description in the normal acquisition mode	13
Fig. 2.3	The generation of over-threshold (OVTH) and shaped trigger (ST) signals for channel <i>i</i> . They are all generated in time with the trigger (Leading Edge Discrimination - LED in figure)	14
Fig. 2.4	Memory organization of 725-730 couples.	14
Fig. 2.5	DPP firmware functional description when the coincidence mode is enabled	15
Fig. 2.6	Generation of the TRG_REQ and TVAW signals for channel <i>i</i> . The T_{TVAW} includes also a latency time T_{LAT} for the signal to go back and forth from mezzanine to mother board.	16
Fig. 2.7	Generation of $ITRG(0) = ITRG(1) = (TRG_REQ(0) \& TRG_REQ(1))$ for channel 0 and channel 1	16
Fig. 2.8	Trigger Management inside couple 0 of 725-730 digitizer series. Couple 0 is made of channel 0 and channel 1. The same applies for the other couples of the 725-730 digitizers.	17
Fig. 2.9	Example of "double coincidence" event	18
Fig. 3.1	Local Trigger Management inside couple 0 of 725-730 digitizer series. Couple 0 is made of channel 0 and channel 1. The same applies for the other couples of the 725-730 digitizers.	26
Fig. 4.1	The "Coincidences" tab under the "Acquisition Setup" window of MC ² Analyzer	39
Fig. 4.2	The "Generic Writes" tab under the "Acquisition Setup" window of MC ² Analyzer	40
Fig. 4.3	MC ² Analyzer configuration of AND between channel 0 and channel 1 with 1.5 μ s of coincidence window.	41
Fig. 4.4	MC ² Analyzer configuration of anti-coincidence between channel 0 and channel 1 with 1.5 μ s of coincidence window.	42
Fig. 4.5	MC ² Analyzer configuration of AND between four channels (from channel 0 to channel 3) with 2.55 μ s of coincidence window.	43
Fig. 4.6	Local Trigger Management inside couple 0 of 725-730 digitizer series. Couple 0 is made of channel 0 and channel 1. The same applies for the other couples of the 725-730 digitizers.	53
Fig. 5.1	Muon Shield scheme: an emitting source is detected by a particular detector. Three scintillators act as a muon shield. Events passing both in the shield and in the detector are removed.	71
Fig. 5.2	Example of two-sided detectors that acquire in coincidence	75
Fig. 5.3	MC ² Analyzer configuration of veto from external trigger	82

List of Tables

Tab. 1.1	List of digitizers/MCA running DPP firmware that support on-line coincidences	7
-----------------	---	---

1 Introduction

There are many applications in physics requiring the measurement of events in coincidence with different detectors. This usually requires the precise measurement of time and delay time. For example the case of particles emitted in cascade from the same nucleus, or the measurement of cosmic rays, where analysing the coincidences can significantly reduce the background. Other examples are the gamma ray and time spectroscopy, the measurement of the absolute activity of a radioactive source, the lifetime measurement, etc.

A coincidence detecting system determines when the time difference between two events is less than a certain time window, known as the resolving time, and saves the corresponding event. There are two methods to make coincidences:

- **Off-line:** all detected events, with their time stamp, are saved and analyzed through dedicated software. The user can write his/her own codes and implement his/her desired requirements for coincidences, by comparing the time stamp information.
- **On-line:** the system is setup to trigger and save coincident events on-line. This method is recommended for high input rates, to reduce the readout throughput.

CAEN digitizers with proper DPP/Default firmware allow to perform both methods.

With the DPP (Digital Pulse Processing) firmware it is possible either to save all the events running the "list mode" option¹ or to enable the on-line coincidences following the instructions of the next sections. In particular the description of this Guide is valid for the DPP modules listed in Tab. 1.1².

Digitizer/MCA	DPP Firmware
720 series	DPP-PSD
724 series	DPP-PHA
725 series	DPP-PSD and DPP-PHA
730 series	DPP-PSD and DPP-PHA
751 series	DPP-PSD
DT5780-N6780	DPP-PHA
DT5781-N6781	DPP-PHA
DT5790	DPP-PSD

Tab. 1.1: List of digitizers/MCA running DPP firmware that support on-line coincidences

In case of Default firmware the user can decide to both save all the events, or to set a global trigger as the majority logic among channels. Please refer to Sec. **Coincidences with Default Firmware** for more details³.

¹Please refer to each specific manual of DPP-PSD [RD1] and DPP-PHA [RD2] for information about saving files in list mode.

²DPP-QDC for 740 series, DPP-ZLE, and DPP-DAW do not support on-line coincidences.

³742 series do not support on-line coincidences.

On-line coincidences: the traditional approach

Consider the case of coincidence between two input pulses (see Fig. 1.1). In the traditional analog approach the two pulses are summed, and the system triggers on the sum pulse, saving only the coincident event. The width τ of the two inputs determines the time window where a coincidence can occur, corresponding to a *resolving time* of 2τ .

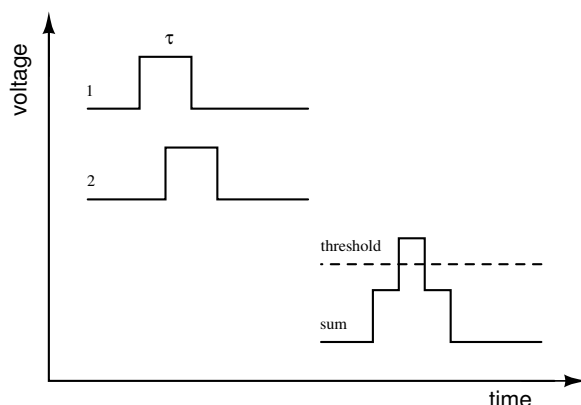


Fig. 1.1: Analog coincidence scheme. The two pulses are summed and the discrimination level is set on the sum pulse, corresponding to the coincidence of the two

The use of timing discriminators, analog-to-digital converters (ADC), and multi-channel analyzers (MCA) allows to trigger on time and obtain a coincidence unit.

There are three ways to perform time trigger, i.e. to generate logic pulses whose leading edge corresponds to the time of occurrence of the input pulse **[RD3]**:

- leading edge;
- crossover;
- constant fraction.

The **leading edge** timing consists on generating a logic pulse when the leading edge of a pulse crosses a fixed discrimination level. Though this is the easiest method, it suffers from the problem of the amplitude walk (the difference in amplitude width that can lead to different time triggers, see Fig. 1.2), and the rise time variation due to possible detector non-uniformities.

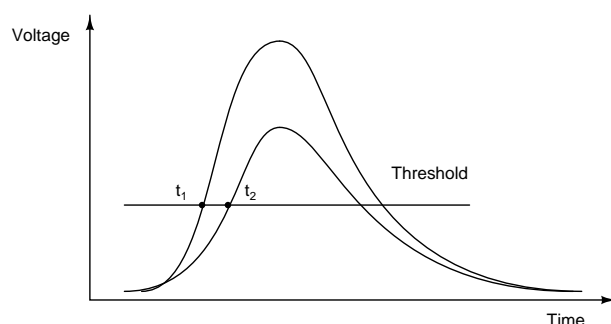


Fig. 1.2: The leading edge timing may suffer from amplitude walk problems

In case of bipolar pulses, the **crossing** timing allows to reduce the amplitude walk issue. Indeed the zero-crossing point (the time when the pulse crosses from the positive to the negative side of the axis) is theoretically independent from the pulse amplitude (see Fig. 1.3).

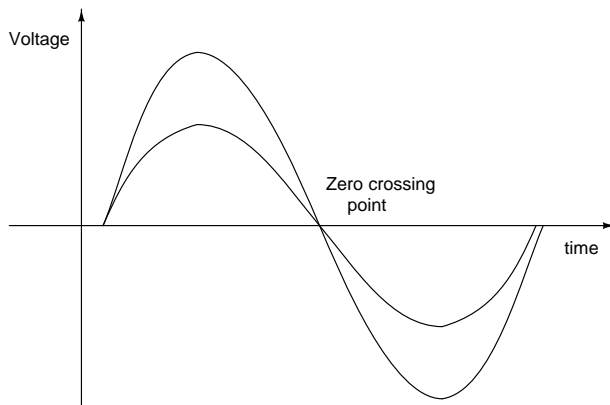


Fig. 1.3: The zero crossing timing of bipolar pulses

The last method consist of triggering on a **constant fraction** of the peak pulse amplitude. For pulses having the same shape, this point is independent on pulse amplitude. The steps to be performed are the following: the input waveform is attenuated by a factor f equal to the desired timing fraction of full amplitude, the signal is inverted and delayed by a time d equal to the time it takes for the pulse to rise from the constant fraction level to the pulse peak; the latest two signals are summed to produce a bipolar pulse. Its zero crossing is at the fraction f of the pulse and it is taken as the time trigger (see Fig. 1.4). The rise time of signals must be equal to apply this method.

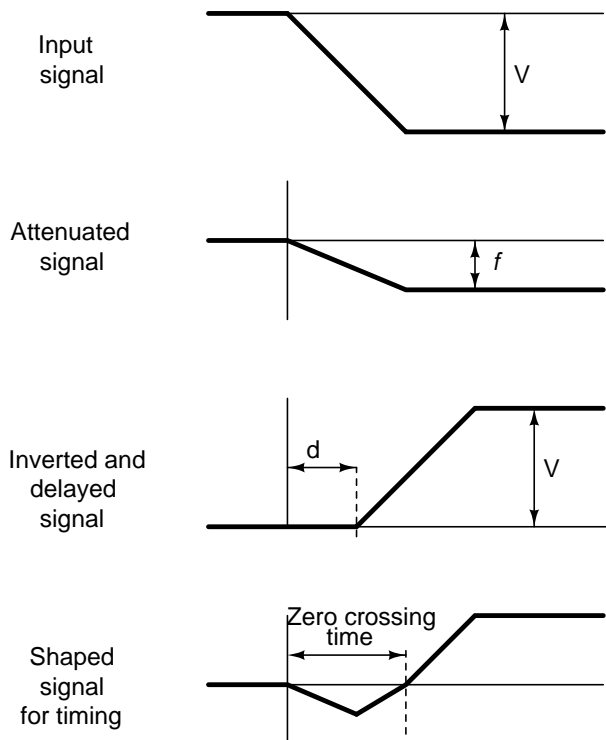


Fig. 1.4: The constant fraction timing method. From top to bottom the input pulse is split into an attenuated signal and a delayed and inverted one. The two are summed and the zero crossing time is taken as the time trigger

A simple system to register coincident events is shown in Fig. 1.5 [RD4].

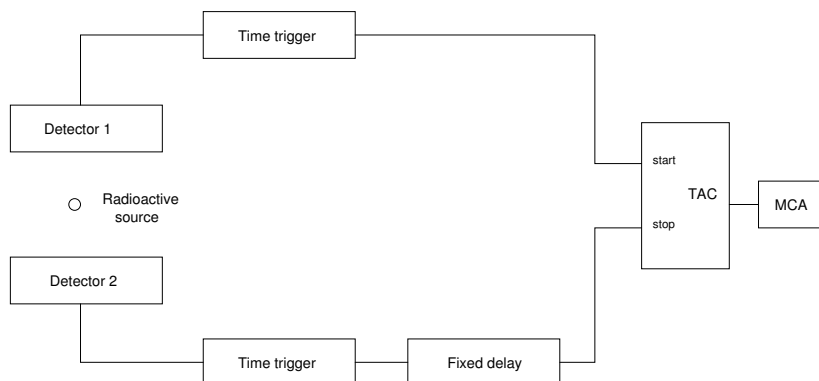


Fig. 1.5: Schematic example showing how to record coincidences of the same event from two different detectors

Suppose that a radioactive source emits events in two detectors at the same time. The events are detected by two timing systems (chosen from one of those previously described) that record their time of detection. One of the two branches is delayed by a fixed amount of time⁴. The two triggers arrive to a time-to-amplitude converted (TAC), a device that produces an output pulse with an amplitude proportional to the time interval between input "start" and "stop" pulses. Finally a multi-channel analyzer (MCA) provides a differential amplitude distribution, also called the "time spectrum".

An example of time spectrum obtained in these conditions is in Fig 1.6. The peak corresponds to the coincidence events, shifted from the axis origin by the same fixed delay of Fig. 1.5. The offset in the y axis corresponds to random events, as for example the noise from detectors that may fake the coincidence. The distribution of the latter events is usually uniform.

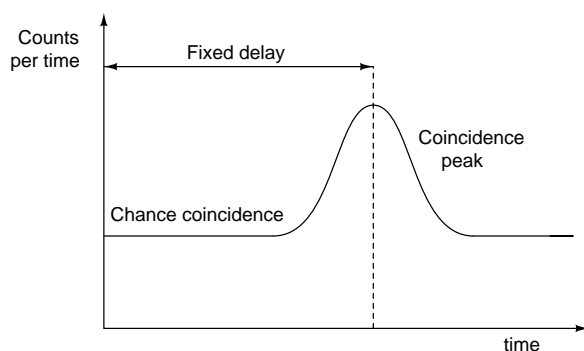


Fig. 1.6: Time spectrum for a radioactive source as in the configuration of Fig. 1.5

⁴A fixed delay time allows to shift the peak of coincidence events to positive values and to correctly evaluate its resolution.

Varying the threshold on the TAC output it is possible to specify the coincidence time window, and adding a variable delay in the top branch of Fig. 1.5 it is possible to obtain the distribution of coincidence rate versus the relative delay between signals, the so called "coincidence-delay curve" (see Fig. 1.7) [RD5]. For a relative delay within the resolving time window, the coincidence rate is equal to the input rate. Outside the resolving time window the coincidence rate is equal to the chance coincidence rate. The transition is sharp for ideal detectors, while real detector goes smoothly to the chance coincidence rate.

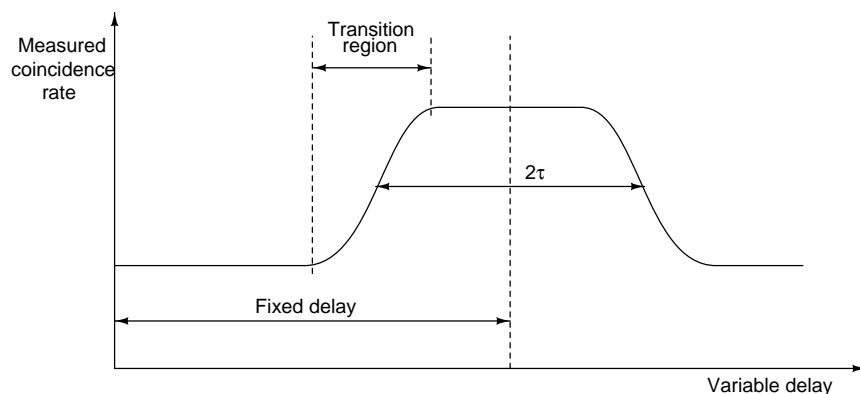


Fig. 1.7: The coincidence rate vs delay curve

2 Coincidences with CAEN DPP firmware

The use of new flash ADCs in acquisition boards gives the possibility to convert the analog signal with improved resolution and faster sampling speed, thus preserving the information required by the physics experiments. The integrated FPGA is then able to acquire the information from the ADCs and process it, through dedicated algorithms. Those algorithms may be the digital replacement of the traditional analog signal processing, so that the digitizer embeds different functions in one single board. In particular, it is possible to replace timing filters such as CFD, Shaper Amplifier, Peak Sensing ADC, QDC, TDC, etc. A single CAEN digitizer therefore is able to replace all modules described in the previous section in a single device (see Fig. 2.1 for a schematic chain).

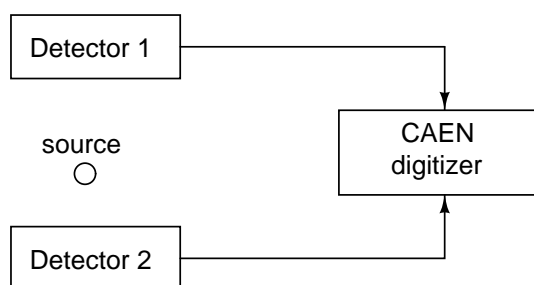


Fig. 2.1: Example of digital chain for coincidences with CAEN digitizer

This is accomplished by two firmware categories: Default firmware, and Digital Pulse Processing (DPP) firmware, which run on-line on the digitizer. While the Default firmware allows the user to record the waveform samples for further analysis, the DPP one is able to analyse the samples on-line and to provide in real time significant information of the event, like energy, time, and pulse shape. The scheme of the DPP firmware functionalities in the "normal" and "coincidence" acquisition modes is described in the next Sections.

DPP Normal Acquisition mode

The normal acquisition mode (coincidence not enabled) for a VME form factor is described in Fig. 2.2. The same considerations are valid also for Desktop (DT) and NIM form factor models, apart for the different number of channels and the LVDS I/O (VME only). Note that TRG-OUT in the VME is called GPO in the DT/NIM. Since the operating scheme is the same for all channels, we are going to describe it only for channel 0.

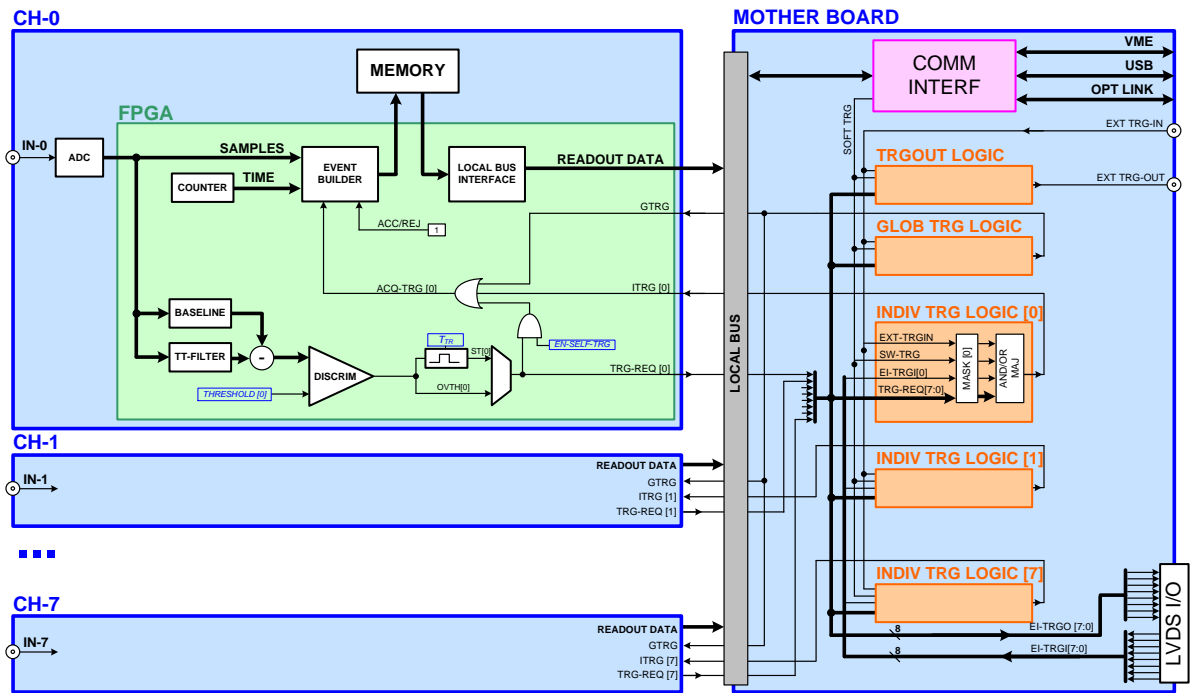


Fig. 2.2: DPP firmware functional description in the normal acquisition mode

The analog input is converted into a digital signal through an ADC, then it enters into the FPGA and it is sent to the "event builder" that evaluates the arrival time, integrates the charge, and performs all DPP specific calculations.

The TT-Filter (Trigger and Timing Filter) may be LED (Leading Edge Discriminator), and CFD (Constant Fraction Discrimination) in case of DPP-PSD firmware **[RD1]**, or RC-CR² in case of DPP-PHA firmware **[RD2]**. After the TT-Filter it is possible to select through a multiplexer the output itself, or a logic pulse of adjustable time width T_{ST} . The latter is the *shaped trigger* that enables the "trigger request" (TRG_REQ) for the data acquisition (bottom branch). The time diagram of the mentioned signals is shown in Fig. 2.3. The same TRG_REQ can also be sent to the mother board for additional operations, as detailed below.

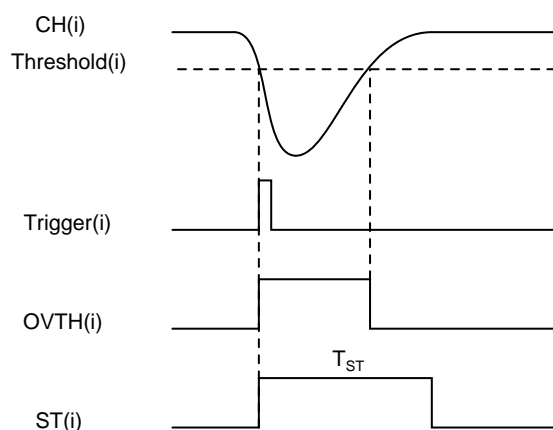


Fig. 2.3: The generation of over-threshold (OVTH) and shaped trigger (ST) signals for channel i. They are all generated in time with the trigger (Leading Edge Discrimination - LED in figure)

In case of 725 and 730 series we must introduce the concept of "couples of channels", where couple n corresponds to channel 2n and channel 2n+1. Channels of the same couple share the same memory SSRAM (see Fig. 2.4) and the same TRG_REQ (see Fig. 2.8). Indeed a single TRG_REQ from the couple is propagated to the mother board FPGA to participate to the coincidence logic.

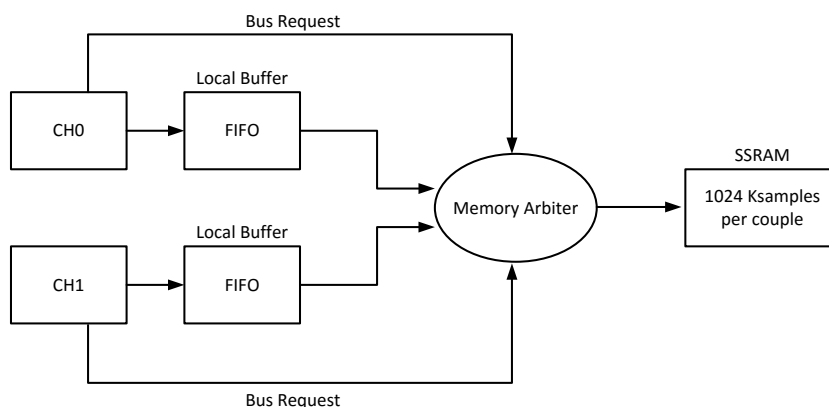


Fig. 2.4: Memory organization of 725-730 couples.

The *event builder* acquires the event and makes all proper calculations as soon as it receives the "acquisition trigger" (ACQ_TRG). Then it waits for the "trigger validation" (TRG_VAL) to write the event in the SSRAM memory. In the normal acquisition mode TRG_VAL is always 1. The ACQ_TRG is the logic OR of the TRG_REQ, the "individual trigger" (ITRG), and the "global trigger" (GTRG).

The ITRG is a logic signal coming from the mother board FPGA through the "individual trigger logic" (ITL), and it has the same multiplicity of the number of channels. Each ITL can receive as input the TRG_REQ from all channels, the signal from line 0 to 7 of the LVDS I/O connector, the software trigger (SOFT_TRG), and the external trigger (EXT TRG-IN). It can perform a set of logic operations: AND, OR, and MAJORITY, where AND/OR are the classic logical operation, and MAJORITY is true when at least a programmable "majority level" number of enabled inputs are in coincidence. The user can set both the majority logic and the majority level. Since there is one ITL for each channel, the output of a single ITL can control only that specific channel.

The "trg-out logic" works as the ITL, but it has no the same multiplicity. Only one TRG-OUT signal is generated for the whole board.

The GTL can receive as inputs the TRG_REQ from all channels, the EXT TRG-IN and the SOFT TRG, while

it cannot receive the inputs from the LVDS I/O. At the moment this logic unit can perform only the OR operation. There is one GTL for all channels, since its output triggers all channels simultaneously.

CAEN digitizers allow the use of different boards simultaneously, as well as the use of external modules to create multi-board systems. For example, it is possible to acquire when both channel X of board N and channel Y of board M detect an input pulse. The trigger request of those channels can be sent out through the output lines (0 to 7) of the LVDS I/O connectors to an external "logic unit" (as the multi-purpose FPGA board V2495 provided by CAEN). The latter implements the logic operations and it sends the TRG_VAL back to the front panel "external trigger in" (EXT TRG-IN) of the VME.

It is possible also to acquire from one channel when other channels from different VME boards fulfil the logic requirements. In this case the input lines (8 to 15) of the LVDS I/O are used to send back the TRG_VAL from the external logic unit.

To perform the last two cases the synchronization among different boards is required. For more details about the synchronization please refer to the specific application note [RD6].

DPP Coincidence mode

In the DPP coincidence mode the acquisition chain is modified as shown in Fig. 2.5.

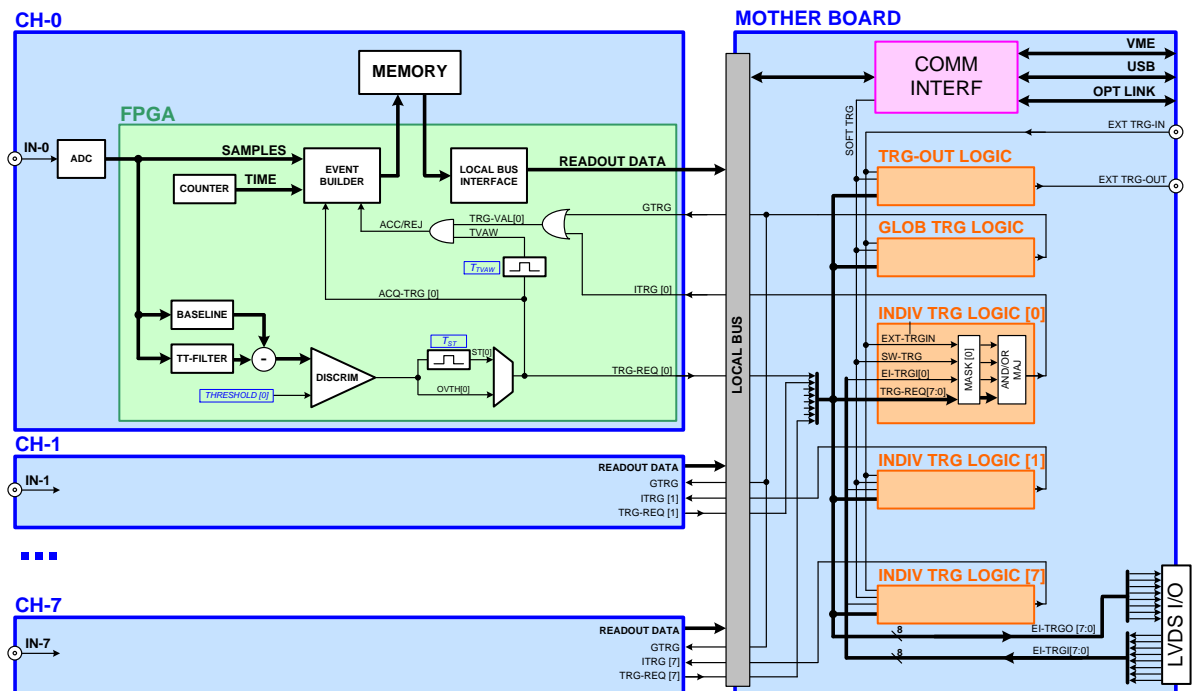


Fig. 2.5: DPP firmware functional description when the coincidence mode is enabled

Differences from the normal acquisition mode (refer to the previous section) lie in the channel FPGA. The ACQ_TRG now corresponds only to the TRG_REQ (i.e. the event is processed whenever a trigger request arrives), while the TRG_VAL is the OR between the GTRG and the ITRG(i). The TRG_VAL can enable the event writing only if it is in time with the "trigger validation acceptance window" (TVAW); when the logic AND between TRG_VAL and TVAW is satisfied the event is saved into the memory. The T_{TVAW} value is set by default as the sum of T_{ST} plus a fixed amount of time that takes into account the signal latency T_{LAT} (see Fig. 2.6).

Fig. 2.7 shows the case where ITRG(0) and ITRG(1) are set as the AND between channel 0 and channel 1 (i.e. $ITRG(0) = ITRG(1) = (TRG_REQ(0) \& TRG_REQ(1))$). In this specific case the ITRGs are in time with the TVAW(i) and the TRG_VAL(i) enables the writing of the i channel into memory.

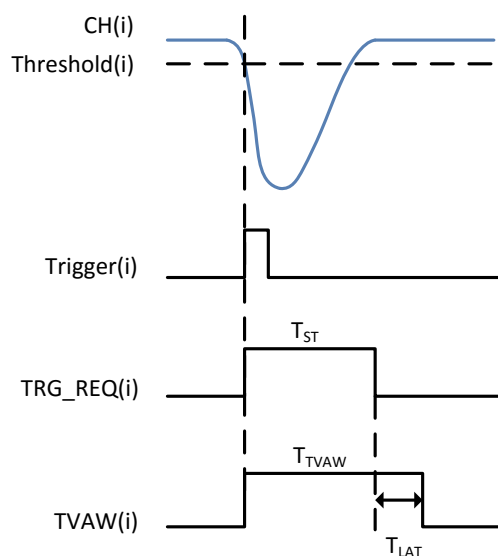


Fig. 2.6: Generation of the TRG_REQ and TVAW signals for channel i . The T_{TVAW} includes also a latency time T_{LAT} for the signal to go back and forth from mezzanine to mother board.

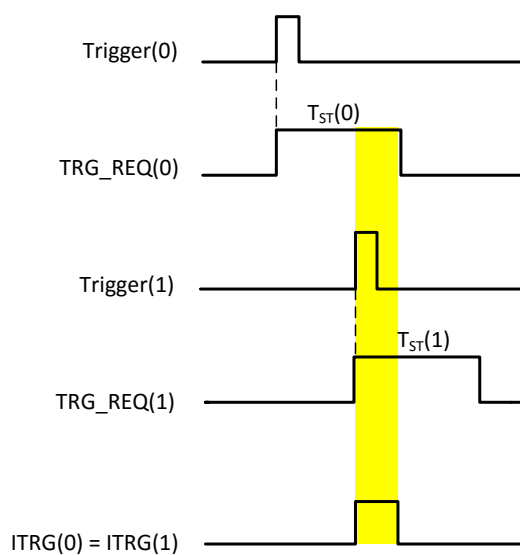


Fig. 2.7: Generation of $ITRG(0) = ITRG(1) = (TRG_REQ(0) \& TRG_REQ(1))$ for channel 0 and channel 1

In case of 725-730 series, only one TRG_REQ per couple is propagated to the mother board to participate to the ITL, GTL, and TRG-OUT Logic. Each channel of the couple generates its local Shaped Trigger, which can be combined to generate the TRG_REQ of the couple. Options are usually AND/OR/one of the channel. One TRG_VAL signal arrives for the couple, where it can be programmed to be the AND/OR/crossed/from mother board (refer to Sec. **How to configure the registers - 725 and 730 series with DPP-PSD** and **How to configure the registers - 725 and 730 series with DPP-PHA**).

A great advantage comes in case of coincidences between channels of the same couple, which can be managed inside the channel FPGA, with no propagation to the mother board. Coincidences among channels of different groups can be performed by means of coincidences among couples, since one TRG_REQ is generated for each couple.

The scheme for 725 and 730 series is shown in Fig. 2.8.

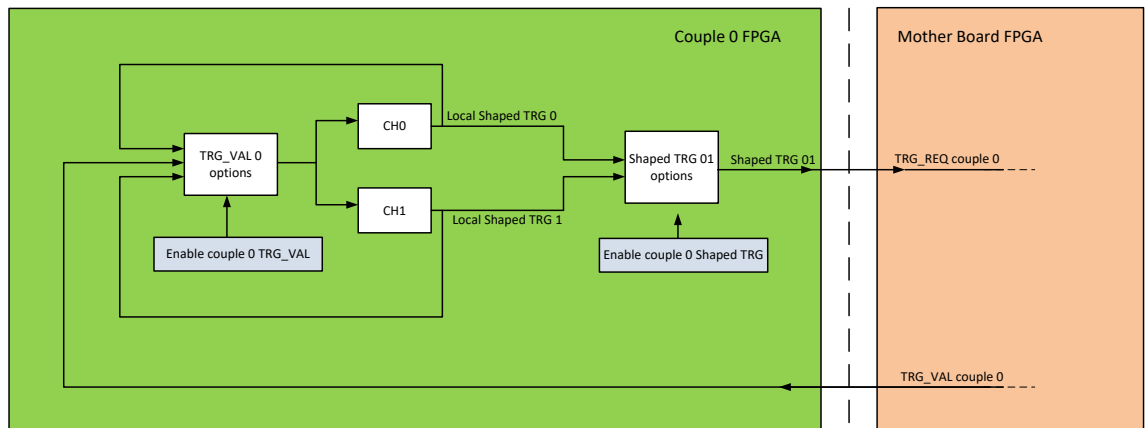


Fig. 2.8: Trigger Management inside couple 0 of 725-730 digitizer series. Couple 0 is made of channel 0 and channel 1. The same applies for the other couples of the 725-730 digitizers.

Resolving time

The resolving time τ (see the definition in Sec. **On-line coincidences: the traditional approach**) is equal to $\tau = 2 \cdot T_{ST} - 1 \text{ clk}$ for DPP-PSD, where 1 clk = 16 ns in case of 725 series, and 8 ns for the others, while $\tau = 2 \cdot T_{ST}$ for DPP-PHA.

Double coincidence

When large coincidence window (i.e. large T_{ST}) are used, it may happen that a "double coincidence" event occurs, where two events of the same channel are in coincidence with a single event of the other channel. Consider for simplicity the case of coincidence between channel 0 and channel 1, where the TRG_VAL(n) signal is set as the AND between the two channels. As can be seen from Fig. 2.9, both the two signals of channel 0 slightly overlap with the single event of channel 1. Two TRG_VAL signals are therefore generated. The first TRG_VAL enables the first signal of channel 0 to be written, as well the signal from channel 1. The second TRG_VAL enables the second signal of channel 0 to be written, but has no effect on channel 1, since it has already received a validation signal.

In this case two events are written for channel 0 and one for channel 1.

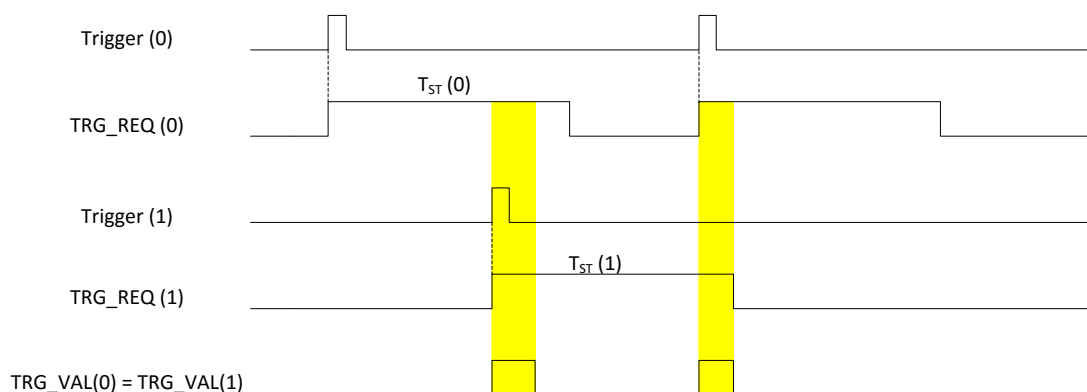


Fig. 2.9: Example of "double coincidence" event

Users must take care to properly setting the coincidence parameters in order to avoid this phenomenon, or, conversely, to take it into account in their off-line analysis. For example, it can be avoided by setting the "Trigger Hold-Off" parameter equal to at least $2 \cdot T_{ST}$.

3 Instructions and Examples for DPP-PSD firmware

This Chapter is intended to describe how to make coincidences with DPP-PSD firmware. The firmware registers involved in the coincidence are described in detail, as well as how to modify the software. In this way those who write their own software, and those who use CAEN software can perform on-line coincidences.

DPP-PSD Control Software

The coincidence mode requires the release version 1.3.3 (or higher) of the DPP-PSD Control Software. For instructions on how to install the DPP-PSD Control Software refer to **[RD1]**.

Registers can be uploaded into the DPP-PSD Control Software through a .txt file called "FreeWrites.txt". Commands on the file are executed at the end of the digitizer programming. The file can be used also to set generic register writes. Further examples of the FreeWrites use can be found in **[RD1]**.

Here follows the instructions for Windows and Linux users.

For **Windows** users, open the DPP-PSD Control Software folder:

```
C:\Program Files\CAEN\Digitizers\DPP-PSD Control Software\data
and follow the instruction written in the file
FreeWritesTemplate.txt
```

```
# Usage:
# 1) Copy this file in the same folder of DPPRunner config
# (see key 'user_config_directory' in file GuiConfig.txt)
# 2) Rename it to 'FreeWrites.txt'
# 3) It is automatically loaded by DppRunner
# at the end of each digitizer's programming
```

For Windows 7, Windows 8, and Windows 10 the DPPRunner folder is a hidden folder located at:

```
C:%HomePath%\AppData\Local\DPP-PSD_ControlSoftware
```



Note: Hidden folder visualization must be prior enabled.

For **Linux** users, open the DPP-PSD Control Software folder:

```
/etc/DPP-PSD_ControlSoftware
and follow the instruction written in the file
FreeWritesTemplate.txt
```

```
# Usage:
# 1) Copy this file in the same folder of DPPRunner config
# (see key 'user_config_directory' in file GuiConfig.txt)
# 2) Rename it to 'FreeWrites.txt'
# 3) It is automatically loaded by DppRunner
# at the end of each digitizer's programming
```

The DPPRunner folder is a hidden folder located at:

```
%Home%\.DPP-PSD_ControlSoftware
```



Note: The FreeWrites.txt file is automatically loaded by the DPP Control Software after the digitizer programming. The digitizer programming is executed for example at the board connection, or any time one of the “DPP Settings” in the GUI (as the “Trigger Threshold”) is changed.

How to configure the registers - 720 and 751 series with DPP-PSD

To configure the coincidence settings it is required to write specific values of the firmware registers **[RD7]**, **[RD8]**. In particular the syntax explained in this section is referred to the “FreeWrites.txt” file syntax **[RD1]**:

```
GENERIC_WRITE 0x<address> 0x<data> 0x<mask>
```

Those who write their own software code, must take care of the proper register write mask, i.e. the bit mask to be written.



Note: all values MUST be written in hexadecimal.

Here the list of registers to be enabled for the coincidences. For more details about the coincidence features refer to Sec. **Coincidences with CAEN DPP firmware**.

- **Enable the propagation of the Individual Trigger (ITRG) from mother board to mezzanines** to perform the signal validation.

Register name	Address	Register bits	Options
Board Configuration	0x8000	[2]	0 ITRG disabled 1 ITRG enabled

Set bit [2] = 1 of the global channel configuration register (Board Configuration), address 0x8000, i.e. write:

```
GENERIC_WRITE 0x8000 0x4 0x4
```

- **Enable the coincidence trigger acquisition mode.** DPP-PSD firmware supports three types of acquisition modes that can be enabled through bits[19:18] of the “DPP Algorithm Control” register. Possible choices are:
 - 00 normal
 - 01 coincidence
 - 11 anti-coincidence

In the *normal acquisition mode*, as described in Sec. **DPP Normal Acquisition mode**, each channel can either self-trigger on the input pulse or acquire from an external trigger. In the *coincidence acquisition mode*, as described in Sec. **DPP Coincidence mode**, each channel self-trigger on the input signal and acquires only when a validation signal arrives within a certain time window. The *anti-coincidence acquisition mode* works in the same way as the coincidence mode, but the internal logic is the opposite, i.e. the validation occurs when no signal arrives in the acceptance time window.

Register name	Address	Register bits	Options
DPP Algorithm Control	0x1n80 (channel n)	[19:18]	00 coincidence disabled 01 coincidence enabled 11 anti-coincidence enabled

Set bits [19:18] = 01 of the channel control register (DPP Algorithm Control). Coincidences must be enabled for each channel involved; for example in case of channel 0 and channel 1 write:

```
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
```

- **Set the coincidence windows** (i.e. the width of the shaped trigger) for the trigger request signal (T_{ST}).

Register name	Address	Register bits	Options
Shaped Trigger Width	0x1n70 (channel n)	[7:0]	n. clocks (hex)

Set the same value for each channel involved in the coincidence. The value is expressed in trigger clock cycles (8 ns for 751 and 720 series), i.e. you have to write the desired value divided by 8 (in hexadecimal). For example, if you want to have a window of 40 ns, you may write:

```
GENERIC_WRITE 0x1070 0x5 0xFF
GENERIC_WRITE 0x1170 0x5 0xFF
```

- **Set the trigger latency** T_{LAT} . This value must be equal to 9 clock cycles (i.e. 72 ns).

```
GENERIC_WRITE 0x106c 0x9 0xFF
GENERIC_WRITE 0x116c 0x9 0xFF
```

- **Write the trigger validation logic**, which corresponds to the logic operation that enables the coincidence. The trigger validation can be generated either by the Individual Trigger Logic, or by the Global Trigger Logic (See Sec. **Coincidences with CAEN DPP firmware**). Register Trigger Validation Mask $0x8180+(4n)$, where n is the channel number, manages the Individual Trigger Logic¹.

It is possible to perform the logic operation between the channels TRG-REQ by modifying bits[9:8], according to the options: OR = 00, AND = 01, and MAJORITY = 10. Bits [12:10] allow to set the majority level m , where for a level m the majority fires when at least $m+1$ trigger requests are high. It is possible also to include the LVDS I/O Global Trigger (bit [28] VME only), LVDS I/O Individual Trigger (bit [29] VME only), the External Trigger (bit [30]), and the Software Trigger (bit [31]) in logic OR with the individual TRG-REQ.

For example, to configure the AND between channel 0 and channel 1, write:

```
GENERIC_WRITE 0x8180 0x103 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x103 0xFFFFFFFF
```

¹Register address corresponds to 0x8180 for channel 0, 0x8184 for channel 1, 0x8188 for channel 2, 0x818C for channel 3, 0x8190 for channel 4, 0x8194 for channel 5, 0x8198 for channel 6, and 0x819C for channel 7

Register name	Address	Register bits	Options
Trigger Validation Mask	0x8180+(4n) (channel n)	[0]	0 TRG_REQ[0] not propagated 1 TRG_REQ[0] propagated
		[1]	0 TRG_REQ[1] not propagated 1 TRG_REQ[1] propagated
		[2]	0 TRG_REQ[2] not propagated 1 TRG_REQ[2] propagated
		[3]	0 TRG_REQ[3] not propagated 1 TRG_REQ[3] propagated
		[4]	0 TRG_REQ[4] not propagated 1 TRG_REQ[4] propagated
		[5]	0 TRG_REQ[5] not propagated 1 TRG_REQ[5] propagated
		[6]	0 TRG_REQ[6] not propagated 1 TRG_REQ[6] propagated
		[7]	0 TRG_REQ[7] not propagated 1 TRG_REQ[7] propagated
		Operation Mask [9:8]	00 OR 01 AND 10 MAJORITY
		Majority Level [12:10]	value for majority
		[27:13]	0 (Reserved)
		LVDS I/O Global Trigger [28]	0 LVDS GTRG not propagated 1 LVDS GTRG propagated
		LVDS I/O Individual Trigger [29]	0 LVDS ITRG not propagated 1 LVDS ITRG propagated
		External Trigger [30]	0 EXT TRG not propagated 1 EXT TRG propagated
		Software Trigger [31]	0 SOFT TRG not propagated 1 SOFT TRG propagated

Examples for 720-751 series

Coincidence between channel 0 and channel 1 (ch0 & ch1)

Channel 0 and channel 1 acquire data within 40 ns of coincidence window:

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0x5 0xFF
GENERIC_WRITE 0x1170 0x5 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
#
GENERIC_WRITE 0x8180 0x103 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x103 0xFFFFFFFF
```

Coincidence among four channels (ch0 & ch1 & ch2 & ch3)

Channel 0 to channel 3 acquire data within 40 ns of coincidence window:

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1380 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0x5 0xFF
GENERIC_WRITE 0x1170 0x5 0xFF
GENERIC_WRITE 0x1270 0x5 0xFF
GENERIC_WRITE 0x1370 0x5 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x136C 0x9 0xFF
#
GENERIC_WRITE 0x8180 0x10F 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x10F 0xFFFFFFFF
GENERIC_WRITE 0x8188 0x10F 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x10F 0xFFFFFFFF
```

Coincidence between channel 0 and channel 1, coincidence between channel 2 and channel 3 (ch0 & ch1), (ch2 & ch3)

Sets the coincidence between couples of channels, channel 0 and channel 1, channel 2 and channel 3 within 40 ns of coincidence window:

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1380 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0x5 0xFF
GENERIC_WRITE 0x1170 0x5 0xFF
GENERIC_WRITE 0x1270 0x5 0xFF
GENERIC_WRITE 0x1370 0x5 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x136C 0x9 0xFF
#
GENERIC_WRITE 0x8180 0x103 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x103 0xFFFFFFFF
GENERIC_WRITE 0x8188 0x10C 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x10C 0xFFFFFFFF
```


Coincidence among eight channels (ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7) - VME only

Channel 0 to channel 7 (VME only) acquire data within 40 ns of coincidence window:

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1380 0x40000 0xC0000
GENERIC_WRITE 0x1480 0x40000 0xC0000
GENERIC_WRITE 0x1580 0x40000 0xC0000
GENERIC_WRITE 0x1680 0x40000 0xC0000
GENERIC_WRITE 0x1780 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0x5 0xFF
GENERIC_WRITE 0x1170 0x5 0xFF
GENERIC_WRITE 0x1270 0x5 0xFF
GENERIC_WRITE 0x1370 0x5 0xFF
GENERIC_WRITE 0x1470 0x5 0xFF
GENERIC_WRITE 0x1570 0x5 0xFF
GENERIC_WRITE 0x1670 0x5 0xFF
GENERIC_WRITE 0x1770 0x5 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x136C 0x9 0xFF
GENERIC_WRITE 0x146C 0x9 0xFF
GENERIC_WRITE 0x156C 0x9 0xFF
GENERIC_WRITE 0x166C 0x9 0xFF
GENERIC_WRITE 0x176C 0x9 0xFF
#
GENERIC_WRITE 0x8180 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8188 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8190 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8194 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8198 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x819C 0x1FF 0xFFFFFFFF
```

How to configure the registers - 725 and 730 series with DPP-PSD

As mentioned in Sec. **DPP Coincidence mode** couples of channels in 725 and 730 series share the same TRG_REQ. The local Shaped Trigger and local Trigger Validation options can be configured through register "DPP Algorithm Control 2", at address 0x1n84 [RD9], and they are summarized in Fig. 3.1. The n index identifies the n-th channel. The register value must be equal for channels of the same couple. Writing the n channel value, it will write the same value in the n+1 channel (channels of the same couple). Refer to [RD9] for more details.

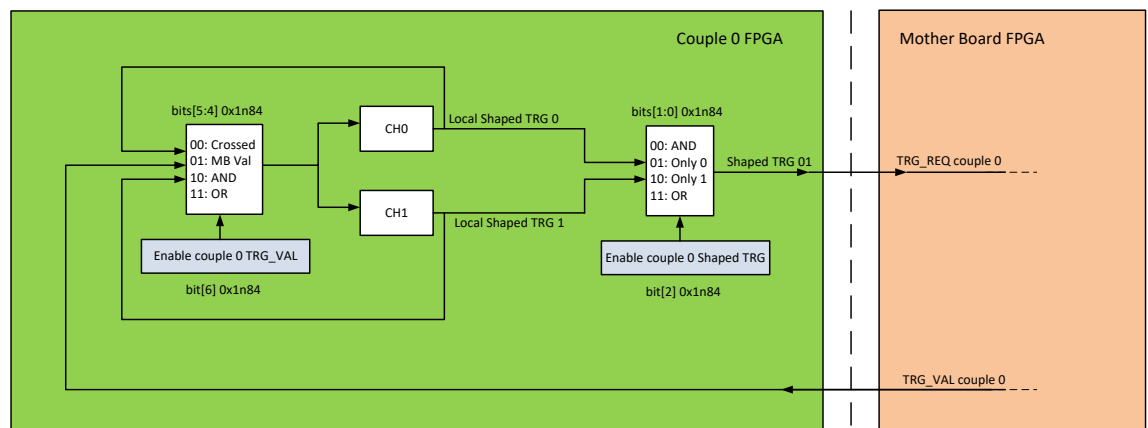


Fig. 3.1: Local Trigger Management inside couple 0 of 725-730 digitizer series. Couple 0 is made of channel 0 and channel 1. The same applies for the other couples of the 725-730 digitizers.

The TRG_REQ of the couple can be the AND, OR of the two channels, or one of the two channels. The TRG_VAL can be generated from mother board, from the other channel of the couple, or it can be the logic AND/OR of the two shaped TRG.

From revision **4.11_136.10** bits[5:4] are modified as shown in the table, and in case of coincidence inside the couple (ch0 & ch1) it is recommended to use option AND (10). The register write is anyhow left unchanged from previous firmware releases.



Note: In the Trigger Validation Mask register, 0x8180(+4n), n is now the couple index, since the validation mask from mother board is referred to the couple, rather than to the single channel.



Note: The coincidence window is expressed in steps of 16 ns for 725 series and 8 ns in case of 730.

To set the coincidence between channel 0 and channel 1 inside the couple (no mother board processing) write:

```
GENERIC_WRITE 0x1084 0x60 0xFF
```

See more examples in the next Sections.

Register name	Address	Register bits	Options
DPP Algorithm Control 2	0x1n84	[1:0]	00: TRG_REQ is the AND of the channels 01: TRG_REQ is the even channel of the couple 10: TRG_REQ is the odd channel of the couple 11: TRG_REQ is the OR of the channels
		[2]	Enable TRG_REQ of the couple
		[3]	Reserved
		[5:4]	00: crossed (TRG_VAL0 = TRG_REQ1; TRG_VAL1 = TRG_REQ0); 01: TRG_VAL0 = TRG_VAL1 = signal from mother-board mask; 10: AND (TRG_VAL0 = TRG_VAL1 = TRG_REQ0 AND TRG_REQ1); 11: OR (TRG_VAL0 = TRG_VAL1 = TRG_REQ0 OR TRG_REQ1).
		[6]	Enable TRG_VAL of the couple

Examples for 725-730 series

Coincidence between channel 0 and channel 1 (ch0 & ch1)

Channel 0 and channel 1 acquire data within 80 ns of coincidence window:

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1170 0xA 0xFF
#
GENERIC_WRITE 0x106C 0x2 0xFF
GENERIC_WRITE 0x116C 0x2 0xFF
#
GENERIC_WRITE 0x1084 0x60 0xFF
```



Note: In case of coincidence inside a couple, set the latency equal to 0x2 (the coincidence is performed in the channel FPGA only with no mother board processing).



Note: In case of 725 write: `GENERIC_WRITE 0x1070 0x5 0xFF` and `GENERIC_WRITE 0x1170 0x5 0xFF` to get 80 ns of coincidence window.

Coincidence inside the couples (ch0 & ch1), (ch2 & ch3), (ch4 & ch5), (ch6 & ch7)

Couples of channels (up to channel 7 for DT/NIM form factors) acquire data within 40 ns of coincidence window:

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>
```

```
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1380 0x40000 0xC0000
GENERIC_WRITE 0x1480 0x40000 0xC0000
GENERIC_WRITE 0x1580 0x40000 0xC0000
GENERIC_WRITE 0x1680 0x40000 0xC0000
GENERIC_WRITE 0x1780 0x40000 0xC0000
```

```
#
```

```
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1170 0xA 0xFF
GENERIC_WRITE 0x1270 0xA 0xFF
GENERIC_WRITE 0x1370 0xA 0xFF
GENERIC_WRITE 0x1470 0xA 0xFF
GENERIC_WRITE 0x1570 0xA 0xFF
GENERIC_WRITE 0x1670 0xA 0xFF
GENERIC_WRITE 0x1770 0xA 0xFF
```

```
#
```

```
GENERIC_WRITE 0x106C 0x2 0xFF
GENERIC_WRITE 0x116C 0x2 0xFF
GENERIC_WRITE 0x126C 0x2 0xFF
GENERIC_WRITE 0x136C 0x2 0xFF
GENERIC_WRITE 0x146C 0x2 0xFF
GENERIC_WRITE 0x156C 0x2 0xFF
GENERIC_WRITE 0x166C 0x2 0xFF
GENERIC_WRITE 0x176C 0x2 0xFF
```

```
#
```

```
GENERIC_WRITE 0x1084 0x60 0xFF
GENERIC_WRITE 0x1284 0x60 0xFF
GENERIC_WRITE 0x1484 0x60 0xFF
GENERIC_WRITE 0x1684 0x60 0xFF
```



Note: In case of coincidence inside a couple, set the latency equal to 0x2 (the coincidence is performed in the channel FPGA only with no mother board processing).



Note: In case of 725 write: `GENERIC_WRITE 0x1070 0x5 0xFF`, etc. to get 80 ns of coincidence window.

Coincidence inside the couples (ch0 & ch1), (ch2 & ch3), (ch4 & ch5), (ch6 & ch7), (ch8 & ch9), (ch10 & ch11), (ch12 & ch13), (ch14 & ch15) - VME only

Couples of channels (up to channel 15 for VME form factor) acquire data within 80 ns of coincidence window:

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1380 0x40000 0xC0000
GENERIC_WRITE 0x1480 0x40000 0xC0000
GENERIC_WRITE 0x1580 0x40000 0xC0000
GENERIC_WRITE 0x1680 0x40000 0xC0000
GENERIC_WRITE 0x1780 0x40000 0xC0000
GENERIC_WRITE 0x1880 0x40000 0xC0000
GENERIC_WRITE 0x1980 0x40000 0xC0000
GENERIC_WRITE 0x1A80 0x40000 0xC0000
GENERIC_WRITE 0x1B80 0x40000 0xC0000
GENERIC_WRITE 0x1C80 0x40000 0xC0000
GENERIC_WRITE 0x1D80 0x40000 0xC0000
GENERIC_WRITE 0x1E80 0x40000 0xC0000
GENERIC_WRITE 0x1F80 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1170 0xA 0xFF
GENERIC_WRITE 0x1270 0xA 0xFF
GENERIC_WRITE 0x1370 0xA 0xFF
GENERIC_WRITE 0x1470 0xA 0xFF
GENERIC_WRITE 0x1570 0xA 0xFF
GENERIC_WRITE 0x1670 0xA 0xFF
GENERIC_WRITE 0x1770 0xA 0xFF
GENERIC_WRITE 0x1870 0xA 0xFF
GENERIC_WRITE 0x1970 0xA 0xFF
GENERIC_WRITE 0x1A70 0xA 0xFF
GENERIC_WRITE 0x1B70 0xA 0xFF
GENERIC_WRITE 0x1C70 0xA 0xFF
GENERIC_WRITE 0x1D70 0xA 0xFF
GENERIC_WRITE 0x1E70 0xA 0xFF
GENERIC_WRITE 0x1F70 0xA 0xFF
#
GENERIC_WRITE 0x106C 0x2 0xFF
GENERIC_WRITE 0x116C 0x2 0xFF
GENERIC_WRITE 0x126C 0x2 0xFF
GENERIC_WRITE 0x136C 0x2 0xFF
GENERIC_WRITE 0x146C 0x2 0xFF
GENERIC_WRITE 0x156C 0x2 0xFF
GENERIC_WRITE 0x166C 0x2 0xFF
GENERIC_WRITE 0x176C 0x2 0xFF
GENERIC_WRITE 0x186C 0x2 0xFF
GENERIC_WRITE 0x196C 0x2 0xFF
GENERIC_WRITE 0x1A6C 0x2 0xFF
GENERIC_WRITE 0x1B6C 0x2 0xFF
GENERIC_WRITE 0x1C6C 0x2 0xFF
GENERIC_WRITE 0x1D6C 0x2 0xFF
```

```

GENERIC_WRITE 0x1E6C 0x2 0xFF
GENERIC_WRITE 0x1F6C 0x2 0xFF
#
GENERIC_WRITE 0x1084 0x60 0xFF
GENERIC_WRITE 0x1284 0x60 0xFF
GENERIC_WRITE 0x1484 0x60 0xFF
GENERIC_WRITE 0x1684 0x60 0xFF
GENERIC_WRITE 0x1884 0x60 0xFF
GENERIC_WRITE 0x1A84 0x60 0xFF
GENERIC_WRITE 0x1C84 0x60 0xFF
GENERIC_WRITE 0x1D84 0x60 0xFF

```



Note: In case of coincidence inside a couple, set the latency equal to 0x2 (the coincidence is performed in the channel FPGA only with no mother board processing).



Note: In case of 725 write: GENERIC_WRITE 0x1070 0x5 0xFF, etc. to get 80 ns of coincidence window.

Coincidence among four channels (four different couples, ch0 & ch2 & ch4 & ch6)

Even channels of the first four couples acquire data within 80 ns of coincidence window. The four couples provide as TRG_REQ only the even channels and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between the four couples.



Note: Since the two channels of the couple share the same memory it is recommended to use only one channel per couple, if available. If it is not possible to use one channel per couple, see the example in the next section.



Note: The odd channels of the couples do not participate to the coincidence logic.

```

# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1480 0x40000 0xC0000
GENERIC_WRITE 0x1680 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1270 0xA 0xFF
GENERIC_WRITE 0x1470 0xA 0xFF
GENERIC_WRITE 0x1670 0xA 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x146C 0x9 0xFF
GENERIC_WRITE 0x166C 0x9 0xFF
#
GENERIC_WRITE 0x1084 0x55 0xFF
GENERIC_WRITE 0x1284 0x55 0xFF
GENERIC_WRITE 0x1484 0x55 0xFF
GENERIC_WRITE 0x1684 0x55 0xFF

```

```
#
GENERIC_WRITE 0x8180 0x10F 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x10F 0xFFFFFFFF
GENERIC_WRITE 0x8188 0x10F 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x10F 0xFFFFFFFF
```



Note: In case of 725 write: `GENERIC_WRITE 0x1070 0x5 0xFF`, etc. to get 80 ns of coincidence window.

Coincidence among four channels (two different couples, ch0 & ch1 & ch2 & ch3)

Channel 0 to channel 3 acquire data within 80 ns of coincidence window. The two couples provide as TRG_REQ the AND of the two channels and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between couple 0 and couple 1.



Note: Since the two channels of the couple share the same memory it is usually suggested to perform the coincidence as in the previous example - if channels are not otherwise occupied.

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1380 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1170 0xA 0xFF
GENERIC_WRITE 0x1270 0xA 0xFF
GENERIC_WRITE 0x1370 0xA 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x136C 0x9 0xFF
#
GENERIC_WRITE 0x1084 0x54 0xFF
GENERIC_WRITE 0x1284 0x54 0xFF
#
GENERIC_WRITE 0x8180 0x103 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x103 0xFFFFFFFF
```



Note: In case of 725 write: `GENERIC_WRITE 0x1070 0x5 0xFF`, etc. to get 80 ns of coincidence window.

Coincidence among eight channels (eight different couples, ch0 & ch2 & ch4 & ch6 & ch8 & ch10 & ch12 & ch14)

Even channels of the eight couples (VME only) acquire data within 80 ns of coincidence window. The couples provide as TRG_REQ only the even channel and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between the eight couples.



Note: Since the two channels of the couple share the same memory it is recommended to use only one channel per couple, if available. If it is not possible to use one channel per couple, see the example in the next section.



Note: The odd channels of the couples do not participate to the coincidence logic.

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1480 0x40000 0xC0000
GENERIC_WRITE 0x1680 0x40000 0xC0000
GENERIC_WRITE 0x1880 0x40000 0xC0000
GENERIC_WRITE 0x1A80 0x40000 0xC0000
GENERIC_WRITE 0x1C80 0x40000 0xC0000
GENERIC_WRITE 0x1E80 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1270 0xA 0xFF
GENERIC_WRITE 0x1470 0xA 0xFF
GENERIC_WRITE 0x1670 0xA 0xFF
GENERIC_WRITE 0x1870 0xA 0xFF
GENERIC_WRITE 0x1A70 0xA 0xFF
GENERIC_WRITE 0x1C70 0xA 0xFF
GENERIC_WRITE 0x1E70 0xA 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x146C 0x9 0xFF
GENERIC_WRITE 0x166C 0x9 0xFF
GENERIC_WRITE 0x186C 0x9 0xFF
GENERIC_WRITE 0x1A6C 0x9 0xFF
GENERIC_WRITE 0x1C6C 0x9 0xFF
GENERIC_WRITE 0x1E6C 0x9 0xFF
#
GENERIC_WRITE 0x1084 0x55 0xFF
GENERIC_WRITE 0x1284 0x55 0xFF
GENERIC_WRITE 0x1484 0x55 0xFF
GENERIC_WRITE 0x1684 0x55 0xFF
GENERIC_WRITE 0x1884 0x55 0xFF
GENERIC_WRITE 0x1A84 0x55 0xFF
GENERIC_WRITE 0x1C84 0x55 0xFF
GENERIC_WRITE 0x1E84 0x55 0xFF
#
GENERIC_WRITE 0x8180 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x1FF 0xFFFFFFFF
```



```

GENERIC_WRITE 0x8188 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8190 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8194 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8198 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x819C 0x1FF 0xFFFFFFFF

```



Note: In case of 725 write: `GENERIC_WRITE 0x1070 0x5 0xFF`, etc. to get 80 ns of coincidence window.

Coincidence among eight channels (four different couples, ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7)

Channel 0 to channel 7 acquire data within 80 ns of coincidence window. The four couples provide as TRG_REQ the AND of the two channels and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between the four couples.



Note: Since the two channels of the couple share the same memory it is usually suggested to perform the coincidence as in the previous example (VME only) - if channels are not otherwise occupied.

```

# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1380 0x40000 0xC0000
GENERIC_WRITE 0x1480 0x40000 0xC0000
GENERIC_WRITE 0x1580 0x40000 0xC0000
GENERIC_WRITE 0x1680 0x40000 0xC0000
GENERIC_WRITE 0x1780 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1170 0xA 0xFF
GENERIC_WRITE 0x1270 0xA 0xFF
GENERIC_WRITE 0x1370 0xA 0xFF
GENERIC_WRITE 0x1470 0xA 0xFF
GENERIC_WRITE 0x1570 0xA 0xFF
GENERIC_WRITE 0x1670 0xA 0xFF
GENERIC_WRITE 0x1770 0xA 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x136C 0x9 0xFF
GENERIC_WRITE 0x146C 0x9 0xFF
GENERIC_WRITE 0x156C 0x9 0xFF
GENERIC_WRITE 0x166C 0x9 0xFF
GENERIC_WRITE 0x176C 0x9 0xFF
#
GENERIC_WRITE 0x1084 0x54 0xFF
GENERIC_WRITE 0x1284 0x54 0xFF

```

```

GENERIC_WRITE 0x1484 0x54 0xFF
GENERIC_WRITE 0x1684 0x54 0xFF
#
GENERIC_WRITE 0x8180 0x10F 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x10F 0xFFFFFFFF
GENERIC_WRITE 0x8188 0x10F 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x10F 0xFFFFFFFF

```



Note: In case of 725 write: `GENERIC_WRITE 0x1070 0x5 0xFF`, etc. to get 80 ns of coincidence window.

Coincidence among sixteen channels (ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7 & ch8 & ch9 & ch10 & ch11 & ch12 & ch13 & ch14 & ch15) - VME only

All channels of a VME board acquire data within 80 ns of coincidence window. The couples provide as TRG_REQ the logic AND of the channels and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between the eight couples.

```

# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1380 0x40000 0xC0000
GENERIC_WRITE 0x1480 0x40000 0xC0000
GENERIC_WRITE 0x1580 0x40000 0xC0000
GENERIC_WRITE 0x1680 0x40000 0xC0000
GENERIC_WRITE 0x1780 0x40000 0xC0000
GENERIC_WRITE 0x1880 0x40000 0xC0000
GENERIC_WRITE 0x1980 0x40000 0xC0000
GENERIC_WRITE 0x1A80 0x40000 0xC0000
GENERIC_WRITE 0x1B80 0x40000 0xC0000
GENERIC_WRITE 0x1C80 0x40000 0xC0000
GENERIC_WRITE 0x1D80 0x40000 0xC0000
GENERIC_WRITE 0x1E80 0x40000 0xC0000
GENERIC_WRITE 0x1F80 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1170 0xA 0xFF
GENERIC_WRITE 0x1270 0xA 0xFF
GENERIC_WRITE 0x1370 0xA 0xFF
GENERIC_WRITE 0x1470 0xA 0xFF
GENERIC_WRITE 0x1570 0xA 0xFF
GENERIC_WRITE 0x1670 0xA 0xFF
GENERIC_WRITE 0x1770 0xA 0xFF
GENERIC_WRITE 0x1870 0xA 0xFF
GENERIC_WRITE 0x1970 0xA 0xFF
GENERIC_WRITE 0x1A70 0xA 0xFF
GENERIC_WRITE 0x1B70 0xA 0xFF
GENERIC_WRITE 0x1C70 0xA 0xFF
GENERIC_WRITE 0x1D70 0xA 0xFF
GENERIC_WRITE 0x1E70 0xA 0xFF

```

```

GENERIC_WRITE 0x1F70 0xA 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x136C 0x9 0xFF
GENERIC_WRITE 0x146C 0x9 0xFF
GENERIC_WRITE 0x156C 0x9 0xFF
GENERIC_WRITE 0x166C 0x9 0xFF
GENERIC_WRITE 0x176C 0x9 0xFF
GENERIC_WRITE 0x186C 0x9 0xFF
GENERIC_WRITE 0x196C 0x9 0xFF
GENERIC_WRITE 0x1A6C 0x9 0xFF
GENERIC_WRITE 0x1B6C 0x9 0xFF
GENERIC_WRITE 0x1C6C 0x9 0xFF
GENERIC_WRITE 0x1D6C 0x9 0xFF
GENERIC_WRITE 0x1E6C 0x9 0xFF
GENERIC_WRITE 0x1F6C 0x9 0xFF
#
GENERIC_WRITE 0x1084 0x54 0xFF
GENERIC_WRITE 0x1284 0x54 0xFF
GENERIC_WRITE 0x1484 0x54 0xFF
GENERIC_WRITE 0x1684 0x54 0xFF
GENERIC_WRITE 0x1884 0x54 0xFF
GENERIC_WRITE 0x1A84 0x54 0xFF
GENERIC_WRITE 0x1C84 0x54 0xFF
GENERIC_WRITE 0x1E84 0x54 0xFF
#
GENERIC_WRITE 0x8180 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8188 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8190 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8194 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x8198 0x1FF 0xFFFFFFFF
GENERIC_WRITE 0x819C 0x1FF 0xFFFFFFFF

```



Note: In case of 725 write: `GENERIC_WRITE 0x1070 0x5 0xFF`, etc. to get 80 ns of coincidence window.

Majority of at least three couples among four couples (eight channels)

The first eight channels of the board acquire when at least three couples of channels are in coincidence within 80 ns (majority level = 2).

Syntax:

GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

```
GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
GENERIC_WRITE 0x1280 0x40000 0xC0000
GENERIC_WRITE 0x1380 0x40000 0xC0000
GENERIC_WRITE 0x1480 0x40000 0xC0000
GENERIC_WRITE 0x1580 0x40000 0xC0000
GENERIC_WRITE 0x1680 0x40000 0xC0000
GENERIC_WRITE 0x1780 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1170 0xA 0xFF
GENERIC_WRITE 0x1270 0xA 0xFF
GENERIC_WRITE 0x1370 0xA 0xFF
GENERIC_WRITE 0x1470 0xA 0xFF
GENERIC_WRITE 0x1570 0xA 0xFF
GENERIC_WRITE 0x1670 0xA 0xFF
GENERIC_WRITE 0x1770 0xA 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x136C 0x9 0xFF
GENERIC_WRITE 0x146C 0x9 0xFF
GENERIC_WRITE 0x156C 0x9 0xFF
GENERIC_WRITE 0x166C 0x9 0xFF
GENERIC_WRITE 0x176C 0x9 0xFF
#
GENERIC_WRITE 0x1084 0x57 0xFF
GENERIC_WRITE 0x1284 0x57 0xFF
GENERIC_WRITE 0x1484 0x57 0xFF
GENERIC_WRITE 0x1684 0x57 0xFF
#
GENERIC_WRITE 0x8180 0xA0F 0xFFFFFFFF
GENERIC_WRITE 0x8184 0xA0F 0xFFFFFFFF
GENERIC_WRITE 0x8188 0xA0F 0xFFFFFFFF
GENERIC_WRITE 0x818C 0xA0F 0xFFFFFFFF
```



Note: To set the majority level = 1 (at least two couples are in coincidence), just modify the last four lines, writing 0x60F rather than 0xA0F



Note: In case of 725 write: GENERIC_WRITE 0x1070 0x5 0xFF, etc. to get 80 ns of coincidence window.

How to configure the registers - DT5790 with DPP-PSD

The register configuration for DT5790 is the same as for the 720 series (refer to Sec. **How to configure the registers - 720 and 751 series with DPP-PSD**), apart for the "Trigger Validation Mask" register, where address 0x8188 corresponds to channel 0, and address 0x818C corresponds to channel 1. Moreover bit[2] must be used for channel 0 and bit[3] must be used for channel 1. Refer to **[RD10]** for additional details.

Register name	Address	Register bits	Options
Trigger Validation Mask	0x8188 (ch0) 0x818C (ch1)	[2]	0 TRG_REQ[0] not propagated 1 TRG_REQ[0] propagated
		[3]	0 TRG_REQ[1] not propagated 1 TRG_REQ[1] propagated
		Operation Mask [9:8]	00 OR 01 AND 10 MAJORITY
		Majority Level [12:10]	value for majority
		[27:13]	0 (Reserved)
		External Trigger [30] Software Trigger [31]	0 EXT TRG not propagated 1 EXT TRG propagated 0 SOFT TRG not propagated 1 SOFT TRG propagated

Examples for DT5790

Coincidence between channel 0 and channel 1 (ch0 & ch1)

Channel 0 and channel 1 acquire data within 40 ns of coincidence window:

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0x40000 0xC0000
GENERIC_WRITE 0x1180 0x40000 0xC0000
#
GENERIC_WRITE 0x1070 0x5 0xFF
GENERIC_WRITE 0x1170 0x5 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
#
GENERIC_WRITE 0x8188 0x10C 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x10C 0xFFFFFFFF
```

Anti-coincidence between channel 0 and channel 1

Channel 0 and channel 1 acquire data in anti-coincidence within 40 ns of time window:

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0xC0000 0xC0000
GENERIC_WRITE 0x1180 0xC0000 0xC0000
#
GENERIC_WRITE 0x1070 0x5 0xFF
GENERIC_WRITE 0x1170 0x5 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
#
GENERIC_WRITE 0x8188 0x10C 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x10C 0xFFFFFFFF
```

4 Instructions and Examples for DPP-PHA firmware

This section is dedicated to digitizers and MCAs running DPP-PHA firmware, as listed in Tab. 1.1. In particular Sec. **MC²Analyzer Software** describes how to set coincidences with the MC²Analyzer Software, while Sec. **How to configure the registers - 724 and 781 series with DPP-PHA**, **How to configure the registers - 780 series with DPP-PHA**, and **How to configure the registers - 725 and 730 series with DPP-PHA** explain the specific registers involved in the configuration.

MC²Analyzer Software

The coincidence mode is managed by the MC²Analyzer Software. For instructions on how to install the MC²Analyzer Software refer to [RD2].

A dedicated tab called "Coincidences" is available under the "Acquisition Setup" window for 724, 780, and 781 series. In case of 725 and 730 series refer to the "Generic Writes" tab (see below). The matrix in the Coincidence tab corresponds to the TRG_VAL mapping for each enabled channel. The i-th row is the TRG_VAL for the i-th channel itself, and each column corresponds to the channel that can participate to the validation signal. Finally letter "E" corresponds to the external trigger.

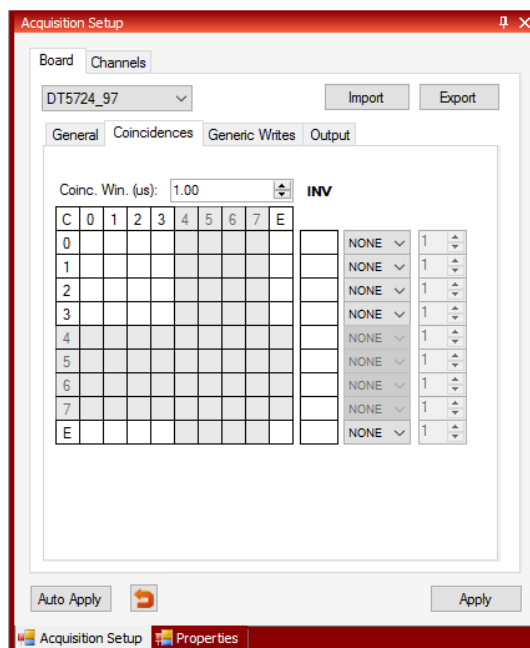


Fig. 4.1: The "Coincidences" tab under the "Acquisition Setup" window of MC²Analyzer

The logic operation can be selected by the scroll-bar menu on the right, choosing among: NONE, AND, OR, MAJ. In case of Majority, it is possible to select the majority level from the menu on the right. The column "INV" allows to select the anti-coincidence mode for each channel.

See the examples in Sec. **Examples with MC²Analyzer Software and 724-780-781 series with DPP-PHA** for more details.



Note: Always press "Apply" to load the configuration, or press "Auto Apply" once to load it automatically.

An additional tab called "Generic Writes" is available to set the register writes and to perform more complex configurations (see Fig. 4.2). In particular in case of 725-730 series, the user must use the Generic Writes tab to set the coincidences.

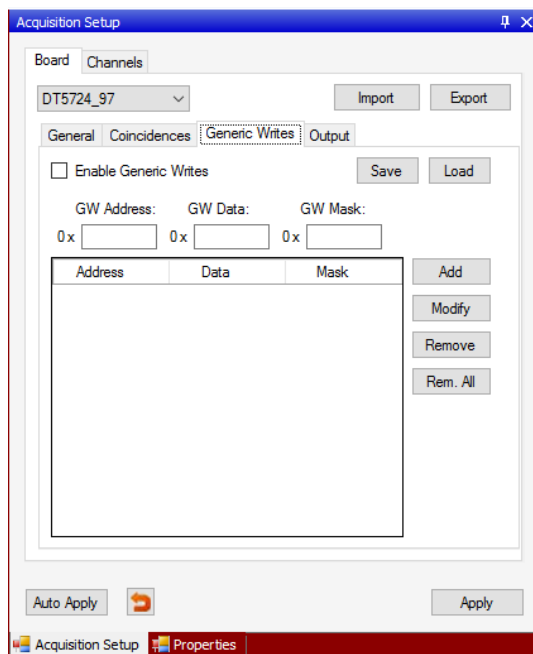
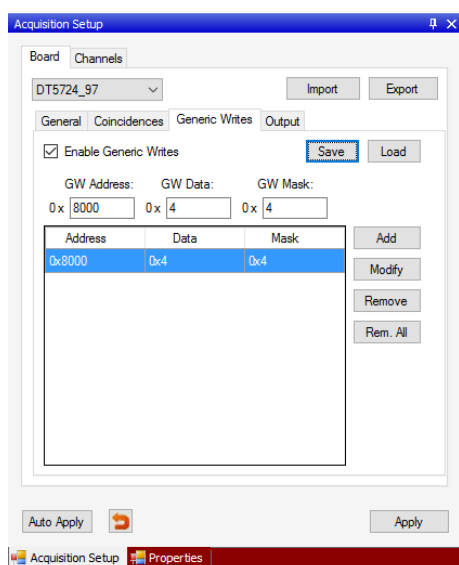


Fig. 4.2: The "Generic Writes" tab under the "Acquisition Setup" window of MC²Analyzer

Click the check-box to enable the register write, then write the register address, the desired value and the mask, i.e. the register bits involved in the write. Buttons "Add", "Modify", "Remove", and "Remove All" allows the user to perform the corresponding operations.

From MC²Analyzer software revision 1.0.10 it is also possible to "Save" a .xml file with the register writes, and to "Load" it. Consider for example the write on the left of the figure below and the corresponding configuration file on the right.



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

See Sec. **How to configure the registers - 724 and 781 series with DPP-PHA**, **How to configure the registers - 780 series with DPP-PHA** and **How to configure the registers - 725 and 730 series with DPP-PHA** for examples about the register writes.

Examples with MC²Analyzer Software and 724-780-781 series with DPP-PHA

Coincidence between channel 0 and channel 1

Channel 0 and channel 1 acquire data within 1.5 μ s of coincidence window. Set the "Coincidences" tab as in Fig. 4.3.

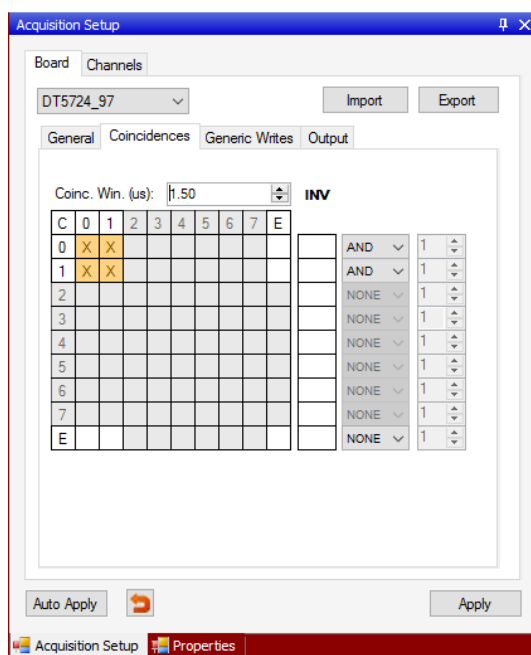


Fig. 4.3: MC²Analyzer configuration of AND between channel 0 and channel 1 with 1.5 μ s of coincidence window.

Anti-coincidence between channel 0 and channel 1

Channel 0 and channel 1 acquire data in anti-coincidence mode within 1.5 μ s of time window. Set the "Coincidences" tab as in Fig. 4.4.

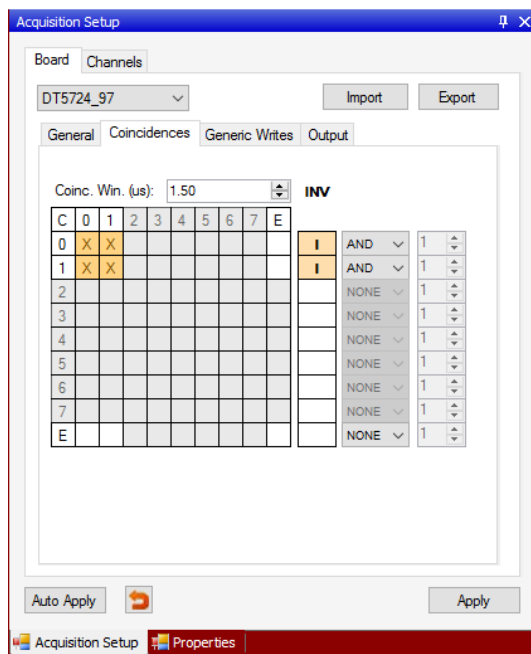


Fig. 4.4: MC² Analyzer configuration of anti-coincidence between channel 0 and channel 1 with 1.5 μ s of coincidence window.

Coincidence among four channels (from channel 0 to channel 3)

Channel 0 to channel 3 acquire data within 2.55 μ s of coincidence window (not available on 780 series). Set the "Coincidences" tab as in Fig. 4.5.

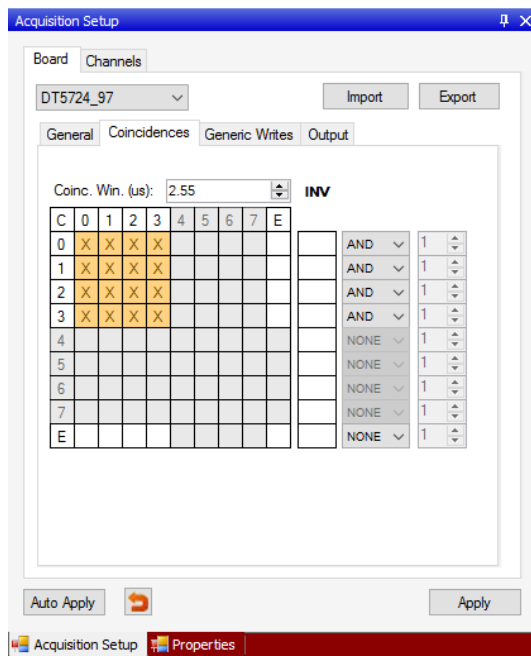


Fig. 4.5: MC² Analyzer configuration of AND between four channels (from channel 0 to channel 3) with 2.55 μ s of coincidence window.

How to configure the registers - 724 and 781 series with DPP-PHA

The register configuration syntax described in this Section makes use of the Generic Write tab of the MC²Analyzer software, as well as of the .xml file generated by the MC²Analyzer software itself. The description can be easily generalized for customer writing their own software, since the bit register involved are mentioned in detail.



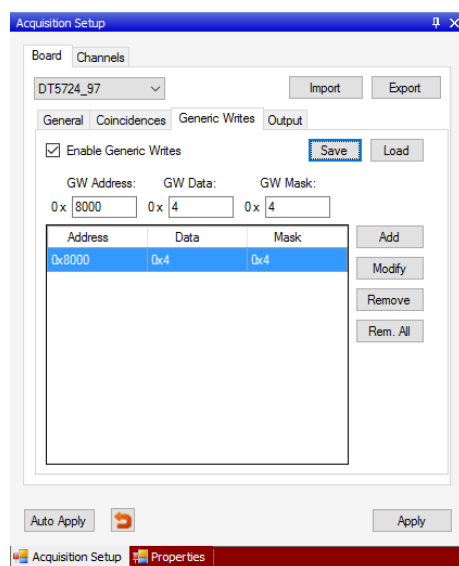
Note: all values MUST be written in hexadecimal.

Here the list of registers to be enabled for the coincidences. For more details about the coincidence features refer to Sec. **Coincidences with CAEN DPP firmware**.

- **Enable the propagation of the Individual Trigger (ITRG) from mother board to mezzanines** to perform the signal validation.

Register name	Address	Register bits	Options
Board Configuration	0x8000	[2]	0 ITRG disabled 1 ITRG enabled

Set bit [2] = 1 of the global channel configuration register (Board Configuration), address 0x8000, i.e. write:



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

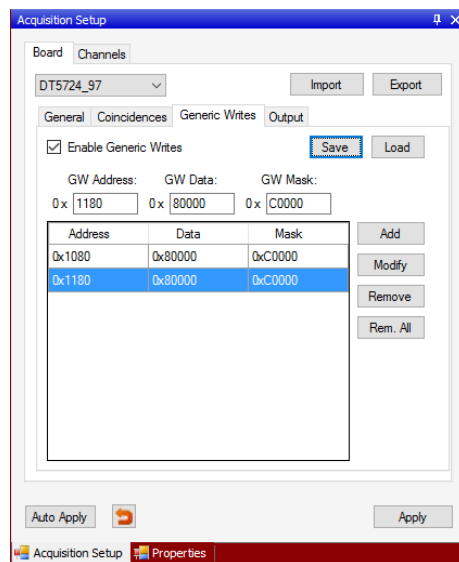
- **Enable the coincidence trigger acquisition mode.** DPP-PHA firmware supports four types of acquisition modes that can be enabled through bits[19:18] of the “DPP Algorithm Control” register. Possible choices are:

00	normal
01	“neighbour”
10	coincidence
11	anti-coincidence

In the *normal acquisition mode*, as described in the **DPP Normal Acquisition mode** section, each channel can either self-trigger on the input pulse or acquire from an external trigger. The “*neighbour*” *acquisition mode* is rather quite useful in case of strip detectors, where you want to read data not only from the channel that triggered, but also from the previous and the consecutive channels (the “neighbours”). Indeed even if a channel is not over-threshold it can receive a TRG_VAL signal as well, if its neighbour fires the trigger. In the *coincidence acquisition mode*, as described in the **DPP Coincidence mode** section, each channel self-trigger on the input signal and acquires only when a validation trigger arrives within a certain time window. The *anti-coincidence acquisition mode* works in the same way of the coincidence mode, but the internal logic is inverted, i.e. the validation occurs when no signal arrives in the acceptance time window. More details can be found in [RD2].

Register name	Address	Register bits	Options
DPP Algorithm Control	0x1n80 (channel n)	[19:18]	00 coincidence disabled 01 “neighbour” enabled 10 coincidence enabled 11 anti-coincidence enabled

To enable the coincidence mode set bits [19:18] = 10 of the “DPP Algorithm Control” register. Coincidences must be enabled for each channel involved; for example in case of channel 0 and channel 1 write:

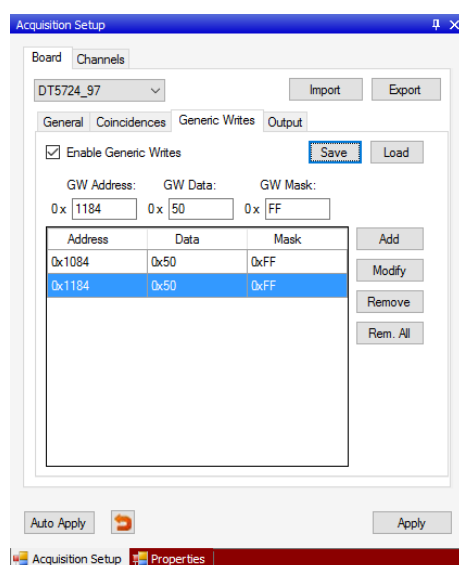


```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>80000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>80000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

- **Set the coincidence windows** (i.e. the width of the shaped trigger) for the trigger request signal (T_{ST}).

Register name	Address	Register bits	Options
Shaped Trigger Width	0x1n84 (channel n)	[7:0]	n. clocks (hex)

Set the same value for each channel involved in the coincidence. The value is expressed in trigger clock cycles (10 ns for 724 series), i.e. you have to write the desired value divided by 10 (in hexadecimal). For example, if you want to have a window of 800 ns, you may write:



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

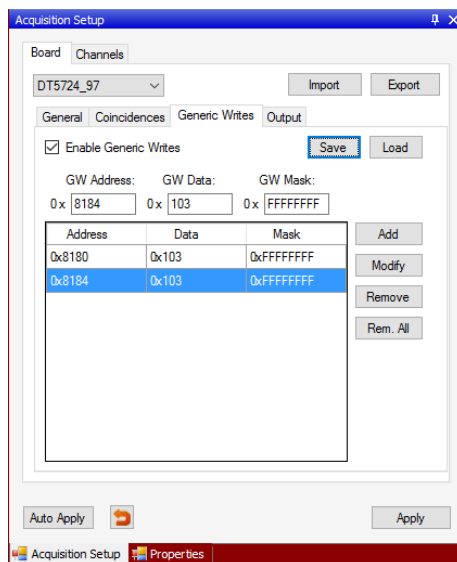
- **Write the trigger validation logic**, which corresponds to the logic operation that enables the coincidence. The trigger validation can be generated either by the Individual Trigger Logic, or by the Global Trigger Logic (See Sec. **Coincidences with CAEN DPP firmware**). Register Trigger Validation Mask 0x8180+(4n), where n is the channel number, manages the Individual Trigger Logic¹.

It is possible to perform the logic operation between the channels TRG-REQ by modifying bits[9:8], according to the options: OR = 00, AND = 01, and MAJORITY = 10. Bits [12:10] allow to set the majority level m , where for a level m the majority fires when at least $m+1$ trigger requests are high. It is possible also to include the LVDS I/O Global Trigger (bit [28] VME only), LVDS I/O Individual Trigger (bit [29] VME only), the External Trigger (bit [30]), and the Software Trigger (bit [31]) in logic OR with the individual TRG-REQ.

For example, to configure the AND between channel 0 and channel 1, write:

¹Register address corresponds to 0x8180 for channel 0, 0x8184 for channel 1, 0x8188 for channel 2, 0x818C for channel 3, 0x8190 for channel 4, 0x8194 for channel 5, 0x8198 for channel 6, and 0x819C for channel 7

Register name	Address	Register bits	Options
Trigger Validation Mask	0x8180+(4n) (channel n)	[0]	0 TRG_REQ[0] not propagated 1 TRG_REQ[0] propagated
		[1]	0 TRG_REQ[1] not propagated 1 TRG_REQ[1] propagated
		[2]	0 TRG_REQ[2] not propagated 1 TRG_REQ[2] propagated
		[3]	0 TRG_REQ[3] not propagated 1 TRG_REQ[3] propagated
		[4]	0 TRG_REQ[4] not propagated 1 TRG_REQ[4] propagated
		[5]	0 TRG_REQ[5] not propagated 1 TRG_REQ[5] propagated
		[6]	0 TRG_REQ[6] not propagated 1 TRG_REQ[6] propagated
		[7]	0 TRG_REQ[7] not propagated 1 TRG_REQ[7] propagated
		Operation Mask [9:8]	00 OR 01 AND 10 MAJORITY
		Majority Level [12:10]	value for majority
		[27:13]	0 (Reserved)
		LVDS I/O Global Trigger [28]	0 LVDS GTRG not propagated 1 LVDS GTRG propagated
		LVDS I/O Individual Trigger [29]	0 LVDS ITRG not propagated 1 LVDS ITRG propagated
		External Trigger [30]	0 EXT TRG not propagated 1 EXT TRG propagated
		Software Trigger [31]	0 SOFT TRG not propagated 1 SOFT TRG propagated

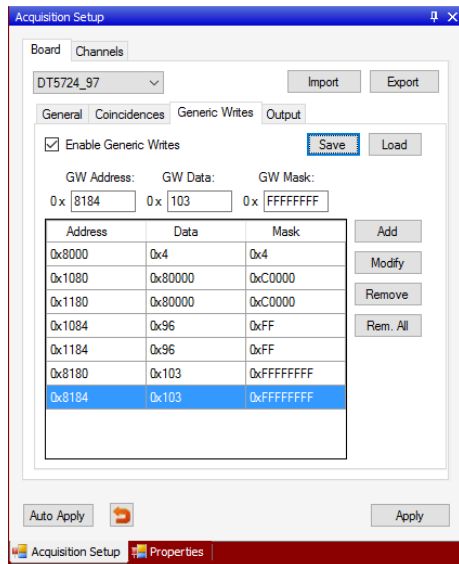


```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>103</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8184</GWaddr>
<GWdata>103</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

Examples for 724 and 781 series with DPP-PHA

Coincidence between channel 0 and channel 1

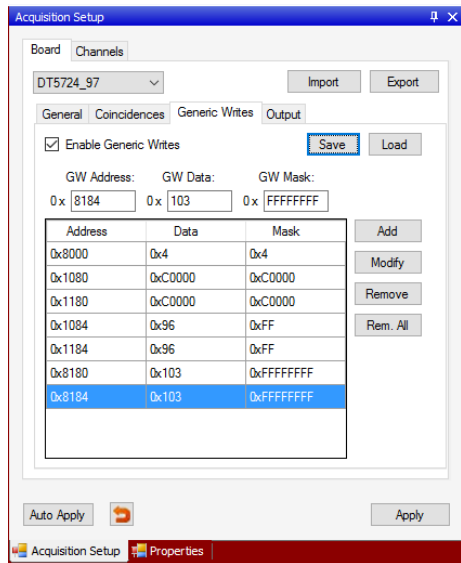
Channel 0 and channel 1 acquire data within 1.5 μ s of coincidence window.



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <GenericWrites>
    <GWEnable>True</GWEnable>
    <GenericWrite>
      <GWaddr>8000</GWaddr>
      <GWdata>4</GWdata>
      <GWmask>4</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1080</GWaddr>
      <GWdata>80000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1180</GWaddr>
      <GWdata>80000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1084</GWaddr>
      <GWdata>96</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1184</GWaddr>
      <GWdata>96</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>8180</GWaddr>
      <GWdata>103</GWdata>
      <GWmask>FFFFFFFF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>8184</GWaddr>
      <GWdata>103</GWdata>
      <GWmask>FFFFFFFF</GWmask>
    </GenericWrite>
  </GenericWrites>
</config>
```


Anti-coincidence between channel 0 and channel 1

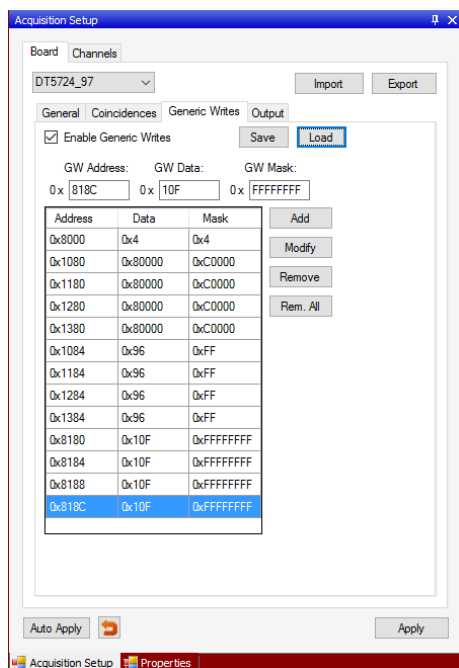
Channel 0 and channel 1 acquire data in anti-coincidence mode within 1.5 μ s of time window.



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <GenericWrites>
    <GWEnable>True</GWEnable>
    <GenericWrite>
      <GWaddr>8000</GWaddr>
      <GWdata>4</GWdata>
      <GWmask>4</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1080</GWaddr>
      <GWdata>C0000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1180</GWaddr>
      <GWdata>C0000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1084</GWaddr>
      <GWdata>96</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1184</GWaddr>
      <GWdata>96</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>8180</GWaddr>
      <GWdata>103</GWdata>
      <GWmask>FFFFFFFF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>8184</GWaddr>
      <GWdata>103</GWdata>
      <GWmask>FFFFFFFF</GWmask>
    </GenericWrite>
  </GenericWrites>
</config>
```

Coincidence among four channels (ch0 & ch1 & ch2 & ch3)

Channel 0 to channel 3 acquire data within 1.5 μ s of coincidence window.



```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
  <GenericWrites>
    <GWEnable>True</GWEnable>
    <GenericWrite>
      <GWaddr>8000</GWaddr>
      <GWdata>4</GWdata>
      <GWmask>4</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1080</GWaddr>
      <GWdata>80000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1180</GWaddr>
      <GWdata>80000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1280</GWaddr>
      <GWdata>80000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1284</GWaddr>
      <GWdata>96</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1384</GWaddr>
      <GWdata>96</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>8180</GWaddr>
      <GWdata>10F</GWdata>
      <GWmask>FFFFFFFF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>8184</GWaddr>
      <GWdata>10F</GWdata>
      <GWmask>FFFFFFFF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>8188</GWaddr>
      <GWdata>10F</GWdata>
      <GWmask>FFFFFFFF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1380</GWaddr>
      <GWdata>80000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1084</GWaddr>
      <GWdata>96</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
  </GenericWrites>
</config>
```

How to configure the registers - 780 series with DPP-PHA

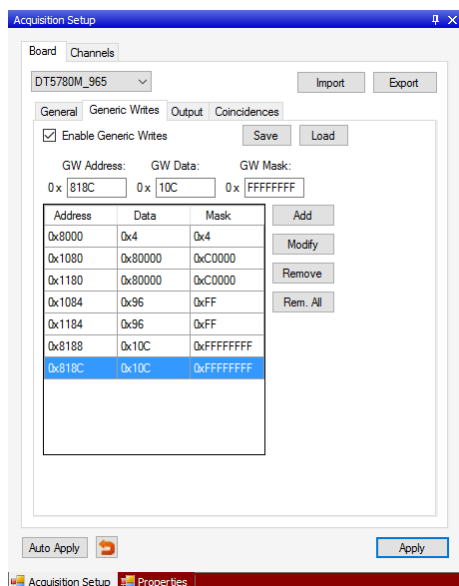
The register configuration for 780 is the same as for the 724 and 781 series (refer to Sec. **How to configure the registers - 724 and 781 series with DPP-PHA**), apart for the "Trigger Validation Mask" register, where address 0x8188 corresponds to channel 0, and address 0x818C corresponds to channel 1. Moreover bit[2] must be used for channel 0 and bit[3] must be used for channel 1.

Register name	Address	Register bits	Options
Trigger Validation Mask	0x8188 (ch0) 0x818C (ch1)	[2]	0 TRG_REQ[0] not propagated 1 TRG_REQ[0] propagated
		[3]	0 TRG_REQ[1] not propagated 1 TRG_REQ[1] propagated
		Operation Mask [9:8]	00 OR 01 AND 10 MAJORITY
		Majority Level [12:10]	value for majority
		[27:13]	0 (Reserved)
		External Trigger [30] Software Trigger [31]	0 EXT TRG not propagated 1 EXT TRG propagated 0 SOFT TRG not propagated 1 SOFT TRG propagated

Examples for 780 series with DPP-PHA

Coincidence between channel 0 and channel 1

Channel 0 and channel 1 acquire data within 1.5 μ s of coincidence window.

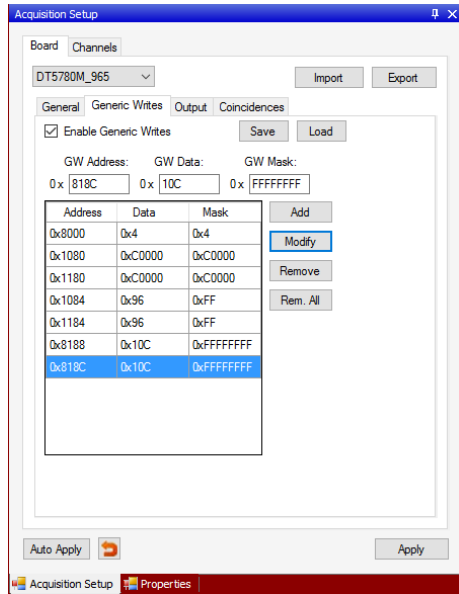


```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>80000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>80000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>96</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>96</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>10C</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>10C</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

```
<GWaddr>1084</GWaddr>
<GWdata>96</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>96</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>10C</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>10C</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

Anti-coincidence between channel 0 and channel 1

Channel 0 and channel 1 acquire data in anti-coincidence mode within 1.5 μ s of time window.



```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>96</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>96</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>10C</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>10C</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

```
<GWaddr>1084</GWaddr>
<GWdata>96</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>96</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>10C</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>10C</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

How to configure the registers - 725 and 730 series with DPP-PHA

In case of 725 and 730 series with DPP-PHA the user must use the register writes to configure the coincidence logic. In particular writes can be performed with the "Generic Writes" tab of the MC²Analyzer software, as explained in Sec. **MC²Analyzer Software**.



Note: all values MUST be written in hexadecimal.

As mentioned in Sec. **DPP Coincidence mode** couples of channels in 725 and 730 series share the same TRG_REQ. The local Shaped Trigger and local Trigger Validation options can be configured through register "DPP Algorithm Control 2", at address 0x1nA0 **[RD11]**, and they are summarized in Fig. 4.6. The n index identifies the n-th channel. The register value must be equal for channels of the same couple. Writing the n channel value, it will write the same value in the n+1 channel (channels of the same couple). Refer to **[RD11]** for more details.

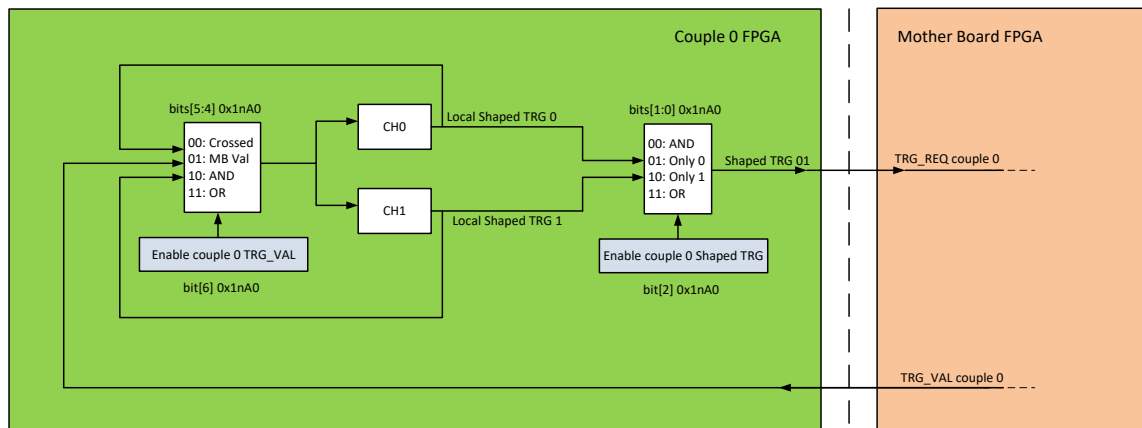


Fig. 4.6: Local Trigger Management inside couple 0 of 725-730 digitizer series. Couple 0 is made of channel 0 and channel 1. The same applies for the other couples of the 725-730 digitizers.

The TRG_REQ of the couple can be the AND, OR of the two channels, or one of the two channels. The TRG_VAL can be generated from mother board, from the other channel of the couple, or it can be the logic AND/OR of the two shaped TRG.

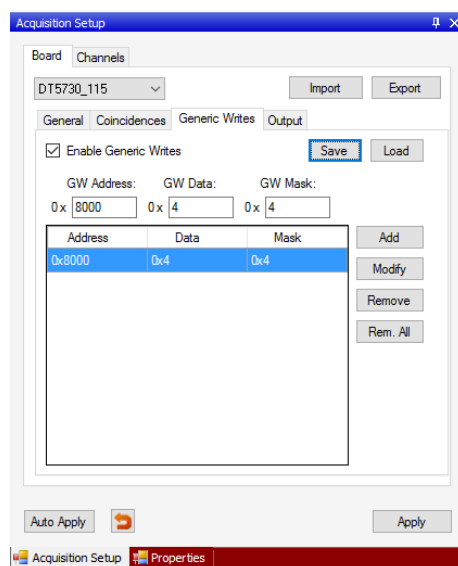
In case of coincidence inside the couple (ch0 & ch1) it is recommended to use option AND (10).

Here the list of registers to be set to enable the coincidence mode.

- **Enable the propagation of the Individual Trigger (ITRG) from mother board to mezzanines** to perform the signal validation.

Register name	Address	Register bits	Options
Board Configuration	0x8000	[2]	0 ITRG disabled 1 ITRG enabled

Set bit [2] = 1 of the global channel configuration register (Board Configuration), address 0x8000, i.e. write:



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

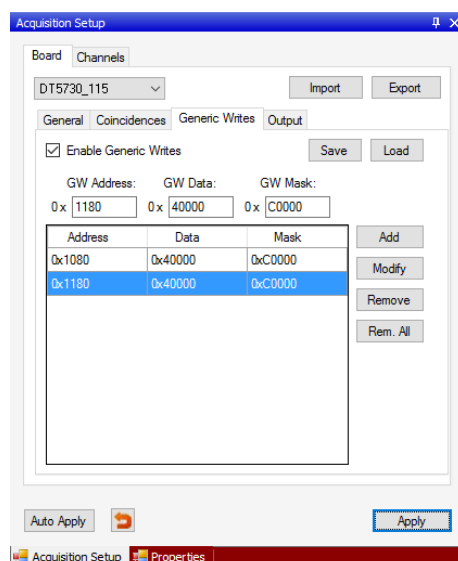
- **Enable the coincidence trigger acquisition mode.** DPP-PHA firmware for 725-730 series supports three types of acquisition modes that can be enabled through bits[19:18] of the “DPP Algorithm Control” register. Possible choices are:

00 normal
01 coincidence
11 anti-coincidence

In the *normal acquisition mode*, as described in Sec. **DPP Normal Acquisition mode**, each channel can either self-trigger on the input pulse or acquire from an external trigger. In the *coincidence acquisition mode*, as described in Sec. **DPP Coincidence mode**, each channel self-trigger on the input signal and acquires only when a validation signal arrives within a certain time window. The *anti-coincidence acquisition mode* works in the same way as the coincidence mode, but the internal logic is the opposite, i.e. the validation occurs when no signal arrives in the acceptance time window.

Register name	Address	Register bits	Options
DPP Algorithm Control	0x1n80 (channel n)	[19:18]	00 coincidence disabled 01 coincidence enabled 11 anti-coincidence enabled

Set bits [19:18] = 01 of the channel control register (DPP Algorithm Control). Coincidences must be enabled for each channel involved; for example in case of channel 0 and channel 1 write:

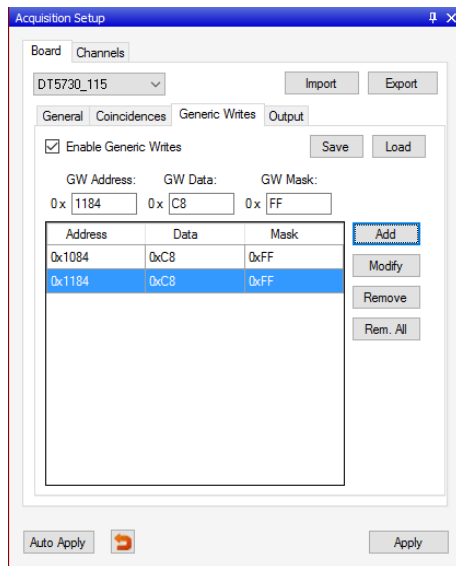


```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

- Set the coincidence windows (i.e. the width of the shaped trigger) for the trigger request signal (T_{ST}).

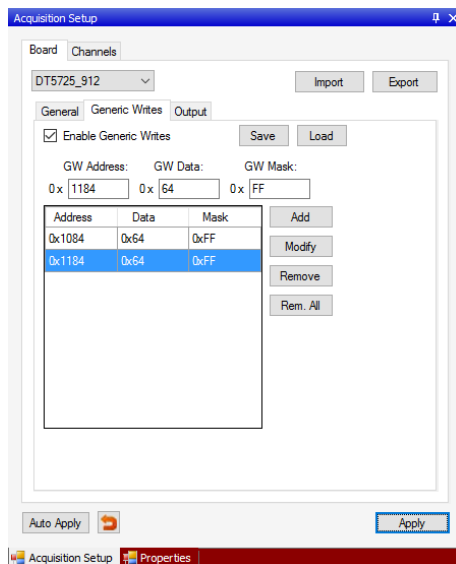
Register name	Address	Register bits	Options
Shaped Trigger Width	0x1n84 (channel n)	[7:0]	n. clocks (hex)

Set the same value for each channel involved in the coincidence. The value is expressed in steps of 16 ns for 725 and 8 ns for 730 series. For example, if you want to have a window of 1.6 μ s, you may write for 730



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

for 725

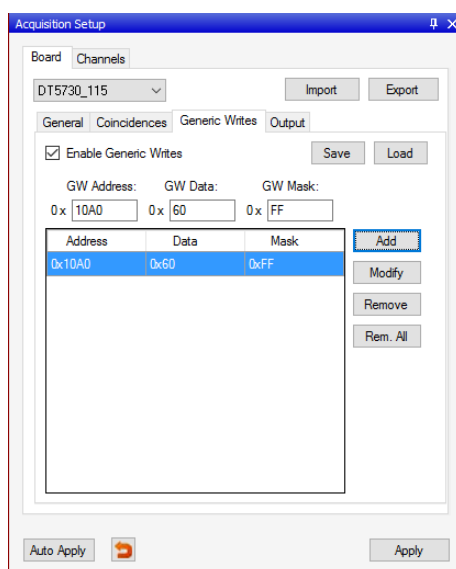


```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>64</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>64</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

- **Set the "DPP Algorithm Control 2" register.** This register manages the local TRG_REQ of the couple (first three bits), and the TRG_VAL origin for the couple itself (bits[6:4]). The TRG_REQ of the couple can be the AND, OR of the two channels, or one of the two channels. The TRG_VAL can be generated from mother board, from the other channel of the couple, or it can be the logic AND/OR of the two possibilities.

Register name	Address	Register bits	Options
DPP Algorithm Control 2	0x1nA0	[1:0]	00: TRG_REQ is the AND of the channels 01: TRG_REQ is the even channel of the couple 10: TRG_REQ is the odd channel of the couple 11: TRG_REQ is the OR of the channels
		[2]	Enable TRG_REQ of the couple
		[3]	Reserved
		[5:4]	00: crossed (TRG_VAL0 = TRG_REQ1; TRG_VAL1 = TRG_REQ0); 01: TRG_VAL0 = TRG_VAL1 = signal from mother-board mask; 10: AND (TRG_VAL0 = TRG_VAL1 = TRG_REQ0 AND TRG_REQ1); 11: OR (TRG_VAL0 = TRG_VAL1 = TRG_REQ0 OR TRG_REQ1).
		[6]	Enable TRG_VAL of the couple

For example, to set the validation from the other channel of the couple (TRG_REQ to mother board not enabled), write:



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>10A0</GWaddr>
<GWdata>60</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

See more examples in the next Section.



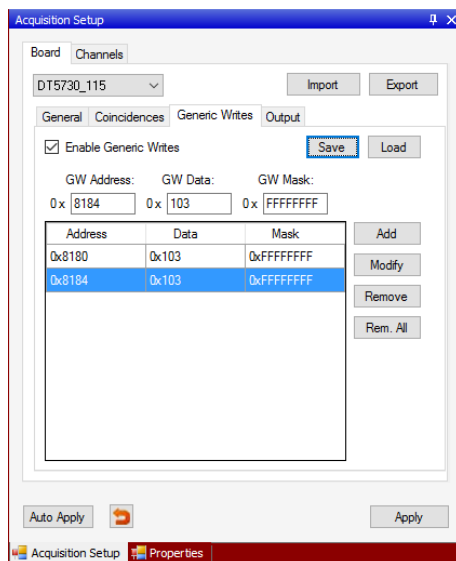
Note: In the address 0x1nA0, n corresponds to the n-th channel. The register value must be equal for channels of the same couple. Writing the 2n channel value, will write the same value in the 2n+1 channel (channels of the same couple). Refer to **[RD11]**.

- **Write the trigger validation logic**, which corresponds to the logic operation that enables the coincidence among couples. The trigger validation can be generated either by the Individual Trigger Logic, or by the Global Trigger Logic (See Sec. **Coincidences with CAEN DPP firmware**). Register Trigger Validation Mask 0x8180+(4n), where n is the couple number, manages the Individual Trigger Logic². It is possible to perform the logic operation between the couples TRG-REQ by modifying bits[9:8], according to the options: OR = 00, AND = 01, and MAJORITY = 10. Bits [12:10] allow to set the majority level *m*, where for a level *m* the majority fires when at least *m*+1 trigger requests are high. It is possible also to include the LVDS I/O Global Trigger (bit [28] VME only), LVDS I/O Individual Trigger (bit [29] VME only), the External Trigger (bit [30],) and the Software Trigger (bit [31]) in logic OR with the individual TRG-REQ.

Register name	Address	Register bits	Options
Trigger Validation Mask	0x8180+(4n) (couple n)	[0]	0 TRG_REQ[0] not propagated 1 TRG_REQ[0] propagated
		[1]	0 TRG_REQ[1] not propagated 1 TRG_REQ[1] propagated
		[2]	0 TRG_REQ[2] not propagated 1 TRG_REQ[2] propagated
		[3]	0 TRG_REQ[3] not propagated 1 TRG_REQ[3] propagated
		[4]	0 TRG_REQ[4] not propagated 1 TRG_REQ[4] propagated
		[5]	0 TRG_REQ[5] not propagated 1 TRG_REQ[5] propagated
		[6]	0 TRG_REQ[6] not propagated 1 TRG_REQ[6] propagated
		[7]	0 TRG_REQ[7] not propagated 1 TRG_REQ[7] propagated
		Operation Mask [9:8]	00 OR 01 AND 10 MAJORITY
		Majority Level [12:10]	value for majority
		[27:13]	0 (Reserved)
		LVDS I/O Global Trigger [28]	0 LVDS GTRG not propagated 1 LVDS GTRG propagated
		LVDS I/O Individual Trigger [29]	0 LVDS ITRG not propagated 1 LVDS ITRG propagated
		External Trigger [30]	0 EXT TRG not propagated 1 EXT TRG propagated
		Software Trigger [31]	0 SOFT TRG not propagated 1 SOFT TRG propagated

²Register address corresponds to 0x8180 for couple 0, 0x8184 for couple 1, 0x8188 for couple 2, 0x818C for couple 3, 0x8190 for couple 4, 0x8194 for couple 5, 0x8198 for couple 6, and 0x819C for couple 7

For example, to configure the AND between couple 0 and couple 1, write:



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>103</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8184</GWaddr>
<GWdata>103</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

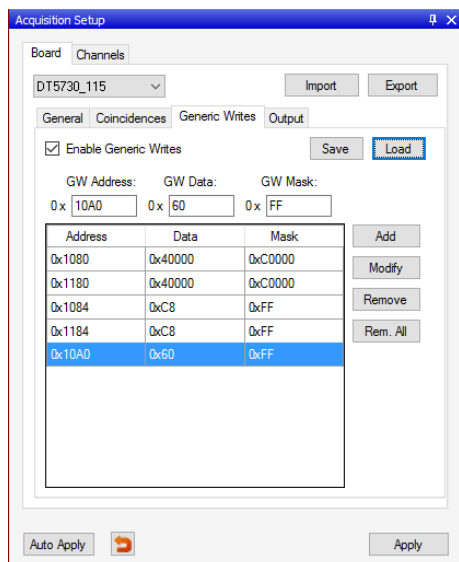
Examples for 725-730 series with DPP-PHA

AND between channel 0 and channel 1 (ch0 & ch1)

Channel 0 and channel 1 acquire data within 1.6 μ s of coincidence window.



Note: In case of 725 write 0x1n84 = 0x64.



```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

```
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>10A0</GWaddr>
<GWdata>60</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

Coincidence inside the couples (ch0 & ch1), (ch2 & ch3), (ch4 & ch5), (ch6 & ch7)

Couples of channels (up to channel 7) acquire data within 1.6 μ s of coincidence window. Since there are many registers involved and their visualization in the Generic Writes tab is not easy, we are going to report the .xml code only. Use the button "Load" to upload the registers in the GUI.



Note: In case of 725 write 0x1n84 = 0x64.

```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
  <GenericWrites>
    <GWEnable>True</GWEnable>
    <GenericWrite>
      <GWaddr>1080</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1180</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1280</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1380</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1480</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1580</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1680</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1780</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1084</GWaddr>
      <GWdata>C8</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1184</GWaddr>
      <GWdata>C8</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1284</GWaddr>
      <GWdata>C8</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1384</GWaddr>
      <GWdata>C8</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1484</GWaddr>
      <GWdata>C8</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1584</GWaddr>
      <GWdata>C8</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1684</GWaddr>
      <GWdata>C8</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1784</GWaddr>
      <GWdata>C8</GWdata>
      <GWmask>FF</GWmask>
    </GenericWrite>
  </GenericWrites>
</config>
```

Coincidence inside the couples (ch0 & ch1), (ch2 & ch3), (ch4 & ch5), (ch6 & ch7), (ch8 & ch9), (ch10 & ch11), (ch12 & ch13), (ch14 & ch15) - VME only

Couples of channels (up to channel 15 for VME form factor) acquire data within 1.6 μ s of coincidence window. Use the button "Load" to upload the registers in the GUI.



Note: In case of 725 write 0x1n84 = 0x64.

```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1280</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1380</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1480</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1580</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1680</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1780</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1880</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1980</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1A80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1B80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1C80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1D80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1E80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1F80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1284</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1384</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1484</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1584</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1684</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1784</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1884</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1984</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1A84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1B84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1C84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1D84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1E84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1F84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
</xml>
```

```

<GMask>FF</GMask>
</GenericWrite>
<GenericWrite>
<GWaddr>1E84</GWaddr>
<GWdata>C8</GWdata>
<GMask>FF</GMask>
</GenericWrite>
<GenericWrite>
<GWaddr>1F84</GWaddr>
<GWdata>C8</GWdata>
<GMask>FF</GMask>
</GenericWrite>
<GenericWrite>
<GWaddr>10A0</GWaddr>
<GWdata>60</GWdata>
<GMask>FF</GMask>
</GenericWrite>
<GenericWrite>
<GWaddr>12A0</GWaddr>
<GWdata>60</GWdata>
<GMask>FF</GMask>
</GenericWrite>
<GenericWrite>
<GWaddr>14A0</GWaddr>
<GWdata>60</GWdata>
<GMask>FF</GMask>
</GenericWrite>
<GenericWrite>
<GWaddr>16A0</GWaddr>
<GWdata>60</GWdata>
<GMask>FF</GMask>
</GenericWrite>
<GenericWrite>
<GWaddr>18A0</GWaddr>
<GWdata>60</GWdata>
<GMask>FF</GMask>
</GenericWrite>
</GenericWrites>
</config>

```

Coincidence among four channels (four different couples, ch0 & ch2 & ch4 & ch6)

Even channels of the first four couples acquire data within 1.6 μ s of coincidence window. The four couples provide as TRG_REQ only the even channels and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between the four couples.



Note: Since the two channels of the couple share the same memory it is recommended to use only one channel per couple, if available. If it is not possible to use one channel per couple, see the example in the next section.



Note: The odd channels of the couples do not participate to the coincidence logic.



Note: In case of 725 write 0x1n84 = 0x64.

Use the button "Load" to upload the registers in the GUI.

```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEEnable>True</GWEEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1280</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1480</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1680</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1284</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1484</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1684</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>10A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>12A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>14A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>16A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

Coincidence among four channels (two different couples, ch0 & ch1 & ch2 & ch3)

Channel 0 to channel 3 acquire data within 1.6 μ s of coincidence window. The two couples provide as TRG_REQ the AND of the two channels and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between couple 0 and couple 1.



Note: Since the two channels of the couple share the same memory it is usually suggested to perform the coincidence as in the previous example - if channels are not otherwise occupied.



Note: In case of 725 write $0x1n84 = 0x64$.

Use the button "Load" to upload the registers in the GUI.

```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1280</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
</GenericWrites>
<GenericWrite>
<GWaddr>1380</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1284</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1384</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

Coincidence among eight channels (eight different couples, ch0 & ch2 & ch4 & ch6 & ch8 & ch10 & ch12 & ch14)

Even channels of the eight couples (VME only) acquire data within 1.6 μ s of coincidence window. The couples provide as TRG_REQ only the even channel and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between the eight couples.



Note: Since the two channels of the couple share the same memory it is recommended to use only one channel per couple, if available. If it is not possible to use one channel per couple, see the example in the next section.



Note: The odd channels of the couples do not participate to the coincidence logic.



Note: In case of 725 write 0x1n84 = 0x64.

Use the button "Load" to upload the registers in the GUI.

```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEEnable>True</GWEEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1280</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1480</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1680</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1880</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1A80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1C80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1E80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>10A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>12A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>14A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>16A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>18A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1AA0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1CA0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1EA0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
<GenericWrite>
<GWaddr>1E84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1E80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1284</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1484</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1684</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1884</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1A84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1C84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
```


</GenericWrite>	</GenericWrite>	</GenericWrite>
<GenericWrite>	<GenericWrite>	<GenericWrite>
<GWaddr>8180</GWaddr>	<GWaddr>818C</GWaddr>	<GWaddr>8198</GWaddr>
<GWdata>1FF</GWdata>	<GWdata>1FF</GWdata>	<GWdata>1FF</GWdata>
<GWmask>FFFFFFFF</GWmask>	<GWmask>FFFFFFFF</GWmask>	<GWmask>FFFFFFFF</GWmask>
</GenericWrite>	</GenericWrite>	</GenericWrite>
<GenericWrite>	<GenericWrite>	<GenericWrite>
<GWaddr>8184</GWaddr>	<GWaddr>8190</GWaddr>	<GWaddr>819C</GWaddr>
<GWdata>1FF</GWdata>	<GWdata>1FF</GWdata>	<GWdata>1FF</GWdata>
<GWmask>FFFFFFFF</GWmask>	<GWmask>FFFFFFFF</GWmask>	<GWmask>FFFFFFFF</GWmask>
</GenericWrite>	</GenericWrite>	</GenericWrite>
<GenericWrite>	<GenericWrite>	</GenericWrites>
<GWaddr>8188</GWaddr>	<GWaddr>8194</GWaddr>	</config>
<GWdata>1FF</GWdata>	<GWdata>1FF</GWdata>	
<GWmask>FFFFFFFF</GWmask>	<GWmask>FFFFFFFF</GWmask>	

Coincidence among eight channels (four different couples, ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7)

Channel 0 to channel 7 acquire data within 1.6 μ s of coincidence window. The four couples provide as TRG_REQ the AND of the two channels and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between the four couples.



Note: Since the two channels of the couple share the same memory it is usually suggested to perform the coincidence as in the previous example (VME only) - if channels are not otherwise occupied.



Note: In case of 725 write $0x1n84 = 0x64$.

Use the button "Load" to upload the registers in the GUI.

```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1280</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1380</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1480</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1580</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1680</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
</GenericWrites>
<GenericWrite>
<GWaddr>1780</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1284</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1384</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1484</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1584</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1684</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1784</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
<GenericWrite>
<GWaddr>10A0</GWaddr>
<GWdata>54</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>12A0</GWaddr>
<GWdata>54</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>14A0</GWaddr>
<GWdata>54</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>16A0</GWaddr>
<GWdata>54</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>10F</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8184</GWaddr>
<GWdata>10F</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>10F</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>10F</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

Coincidence among sixteen channels (ch0 & ch1 & ch2 & ch3 & ch4 & ch5 & ch6 & ch7 & ch8 & ch9 & ch10 & ch11 & ch12 & ch13 & ch14 & ch15) - VME only

All channels of a VME board acquire data within 1.6 μ s of coincidence window. The four couples provide as TRG_REQ the AND of the two channels and receive the TRG_VAL from mother board. The mother board FPGA performs the AND between the four couples.



Note: In case of 725 write 0x1n84 = 0x64.

Use the button "Load" to upload the registers in the GUI.

```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1280</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1380</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1480</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1580</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1680</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1780</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1880</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1980</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1A80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1B80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1C80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1D80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1E80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1F80</GWaddr>
<GWdata>40000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1284</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1384</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1484</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1584</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1684</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1784</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1884</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1984</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1A84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
```

```
<Gwaddr>8184</Gwaddr>
<Gwdata>1FF</Gwdata>
<Gwmask>FFFFFFFF</Gwmask>
</GenericWrite>
<GenericWrite>
<Gwaddr>8188</Gwaddr>
<Gwdata>1FF</Gwdata>
<Gwmask>FFFFFFFF</Gwmask>
</GenericWrite>
<GenericWrite>
<Gwaddr>818C</Gwaddr>
<Gwdata>1FF</Gwdata>
<Gwmask>FFFFFFFF</Gwmask>
</GenericWrite>
<GenericWrite>
<Gwaddr>8190</Gwaddr>
<Gwdata>1FF</Gwdata>
<Gwmask>FFFFFFFF</Gwmask>
</GenericWrite>
<GenericWrite>
<Gwaddr>8194</Gwaddr>
<Gwdata>1FF</Gwdata>
<Gwmask>FFFFFFFF</Gwmask>
</GenericWrite>
<GenericWrite>
<Gwaddr>8198</Gwaddr>
<Gwdata>1FF</Gwdata>
<Gwmask>FFFFFFFF</Gwmask>
</GenericWrite>
<GenericWrite>
<Gwaddr>819C</Gwaddr>
<Gwdata>1FF</Gwdata>
<Gwmask>FFFFFFFF</Gwmask>
</GenericWrite>
</GenericWrites>
</config>
```

Majority of at least three couples among four couples (eight channels)

The first eight channels of the board acquire when at least three couples are in coincidence within 1.6 μ s (majority level = 2).



Note: To set the majority level = 1 (at least two couples are in coincidence), just modify the last four registers, writing 0x60F rather than 0xA0F



Note: In case of 725 write 0x1n84 = 0x64.

Use the button "Load" to upload the registers in the GUI.

```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
  <GenericWrites>
    <GWEnable>True</GWEnable>
    <GenericWrite>
      <GWaddr>8000</GWaddr>
      <GWdata>4</GWdata>
      <GWmask>4</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1080</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1180</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1280</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1380</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1480</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1580</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
    <GenericWrite>
      <GWaddr>1680</GWaddr>
      <GWdata>40000</GWdata>
      <GWmask>C0000</GWmask>
    </GenericWrite>
  </GenericWrites>
  <GenericWrite>
    <GWaddr>1780</GWaddr>
    <GWdata>40000</GWdata>
    <GWmask>C0000</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>1084</GWaddr>
    <GWdata>C8</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>1184</GWaddr>
    <GWdata>C8</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>1284</GWaddr>
    <GWdata>C8</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>1384</GWaddr>
    <GWdata>C8</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>1484</GWaddr>
    <GWdata>C8</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>1584</GWaddr>
    <GWdata>C8</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>1684</GWaddr>
    <GWdata>C8</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>1784</GWaddr>
    <GWdata>C8</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>10A0</GWaddr>
    <GWdata>57</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>12A0</GWaddr>
    <GWdata>57</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>14A0</GWaddr>
    <GWdata>57</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>16A0</GWaddr>
    <GWdata>57</GWdata>
    <GWmask>FF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>8180</GWaddr>
    <GWdata>A0F</GWdata>
    <GWmask>FFFFFFF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>8184</GWaddr>
    <GWdata>A0F</GWdata>
    <GWmask>FFFFFFF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>8188</GWaddr>
    <GWdata>A0F</GWdata>
    <GWmask>FFFFFFF</GWmask>
  </GenericWrite>
  <GenericWrite>
    <GWaddr>818C</GWaddr>
    <GWdata>A0F</GWdata>
    <GWmask>FFFFFFF</GWmask>
  </GenericWrite>
</config>
```

5 How to Veto the acquisition

In many physics applications it is required to veto the acquisition of one or more channels for a certain amount of time. There are various sources of veto, which can be a combination of the other channels, or an external signal. Consider for example the case of a muon shield, or the case of In the following sections there are reported three case:

1. one channel is vetoed by the other enabled channels;
2. taking advantage of the couple management of 725-730 series, it is possible to set a coincidence logic for the channels in the couple and a veto which is managed by the mother board FPGA;
3. an external signal is used to veto the acquisition.

Signal on channel 0 vetoed by the other channels

This is for example the case of a muon shield, where the source is detected by a specific detector (channel 0), and a set of three scintillators (from channel 1 to channel 3) act as a veto (see Fig. 5.1). If a muon crosses one of the three scintillators of the shield and the detector, that event is removed.

The logic is $\overline{CH0} \& (CH1 | CH2 | CH3)$, i.e. the detector acquire in anti-coincidence with the OR among the three scintillators.

In the following subsection the register settings for each firmware are reported.

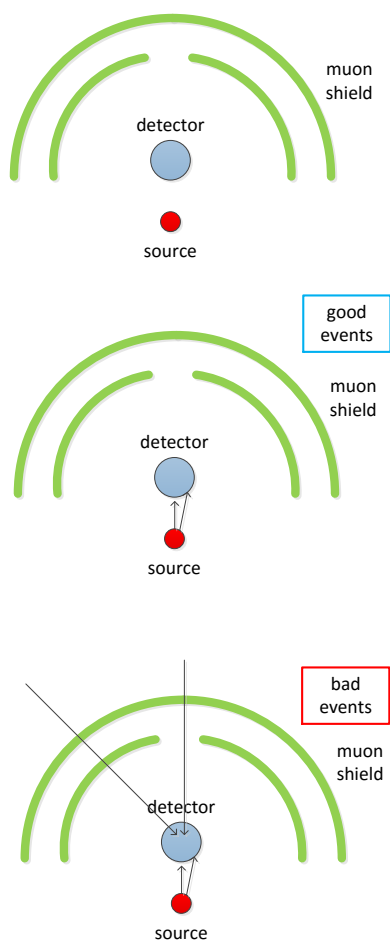


Fig. 5.1: Muon Shield scheme: an emitting source is detected by a particular detector. Three scintillators act as a muon shield. Events passing both in the shield and in the detector are removed.

Veto on ch0 - DPP-PSD for 720 and 751 series

Write the following code in the FreeWrites file. The coincidence window is set to 1 μ s.

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4

GENERIC_WRITE 0x1080 0xC0000 0xC0000

GENERIC_WRITE 0x1070 0x7D 0xFF

GENERIC_WRITE 0x106C 0x9 0xFF

GENERIC_WRITE 0x8180 0xE 0xFFFFFFFF
```

Veto on ch0 - DPP-PSD for 725 and 730 series

Write the following code in the FreeWrites file. The coincidence window is set to 1 μ s.



Note: In case of 725 write: GENERIC_WRITE 0x1070 0x3E 0xFF to get about 1 μ s of coincidence window.

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4

GENERIC_WRITE 0x1080 0xC0000 0xC0000

GENERIC_WRITE 0x1084 0x55 0xFF
GENERIC_WRITE 0x1284 0x5 0xFF
GENERIC_WRITE 0x1484 0x5 0xFF
GENERIC_WRITE 0x1684 0x5 0xFF

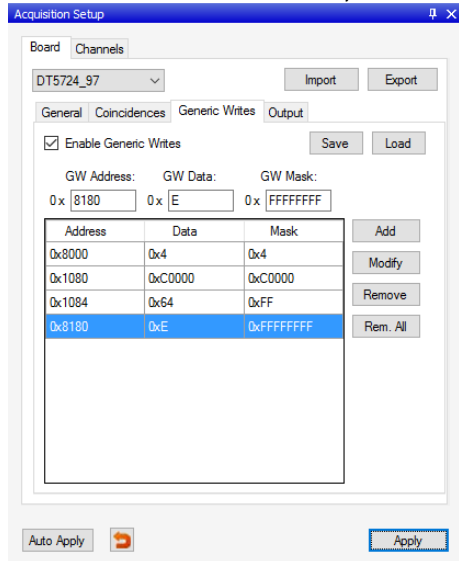
GENERIC_WRITE 0x1070 0x7D 0xFF

GENERIC_WRITE 0x106C 0x9 0xFF

GENERIC_WRITE 0x8180 0xE 0xFFFFFFFF
```


Veto on ch0 - DPP-PHA for 724 and 781 series

Copy the following settings in the Generic Writes tab of the MC²Analyzer software, or load the file. The coincidence window is set to 1 μ s.



```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
```

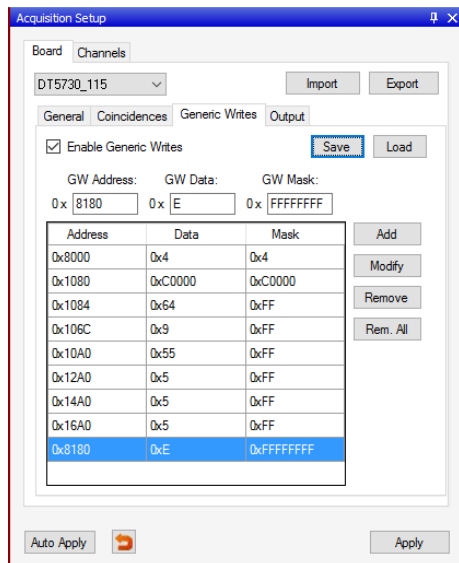
```
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>64</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>E</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

Veto on ch0 - DPP-PHA for 725 and 730 series

Copy the following settings in the Generic Writes tab of the MC²Analyzer software, or load the file. The coincidence window is set to 1 μ s.



Note: In case of 725 write: GENERIC_WRITE 0x1070 0x3E 0xFF to get about 1 μ s of coincidence window.



```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>64</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>106C</GWaddr>
<GWdata>9</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>10A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>12A0</GWaddr>
<GWdata>5</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>14A0</GWaddr>
<GWdata>5</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>16A0</GWaddr>
<GWdata>5</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>E</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

```
<GWaddr>10A0</GWaddr>
<GWdata>55</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>12A0</GWaddr>
<GWdata>5</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>14A0</GWaddr>
<GWdata>5</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>16A0</GWaddr>
<GWdata>5</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>E</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

Veto for the couples of channels (725-730 only)

In case of 725 and 730 series, we can take advantage of the couple management by setting a coincidence logic for the channels in the couple (mezzanine level), and a veto which is controlled by the mother board FPGA (refer to Sec. **Coincidences with CAEN DPP firmware** for more details on how the coincidence is managed in the digitizer).

The source of this veto is managed by bits[19:18] of register 0x1n84 for DPP-PSD, and bits[15:14] of register 0x1nA0 for DPP-PHA.

Register name	Address	Register bits	Options
DPP Algorithm Control 2	0x1n84-PSD, 0x1nA0-PHA	[19:18] ([15:14])	00 = disabled; 01 = the veto signal is common among all channels. It can be set through register 0x810C, and it can be generated by an external trigger or by a combination of the trigger requests from couples; 10 = the veto signal is individually set for the couple of channels (each couple can have a different veto). It can be set through register 0x8180 (+4n), where n is the couple index, and it can be generated by an external trigger or by a combination of the trigger requests from couples; 11 = the veto signal comes from negative saturation.



Note: from revision 136.9 of the DPP-PSD firmware the former bit[7] option has been discontinued. Use option "10" for the same functionality.

The veto duration is defined by register 0x1nD4, where zero means that the veto lasts for the duration of the signal that generated it. A veto width different from 0 extends the veto duration by the amount of time written in the register.

Register name	Address	Register bits	Options
Veto Width	0x1nD4	[15:0]	Value for the veto extension.
		[17:16]	Steps of the veto width. 00: 16 ns for 725, 8 ns for 730; 01: 4 μ s for 725, 2 μ s for 730; 10: 1048 μ s for 725, 524 μ s for 730; 11: 264 ms for 725, 134 ms for 730.

Coincidence inside the VME couples (ch-i & ch-i+1) vetoed by the majority of two or more couples (DPP-PSD)

Couples of channels (up to channel 15 for VME form factor) acquire data within 80 ns of coincidence window. The acquisition is vetoed when at least two couples are in coincidence. For example, this is the case of two-faced detectors which acquire in coincidence. The acquisition is then vetoed by a cosmic ray that crosses two or more couples (see figure below as a scheme).



Note: In case of 725 write: GENERIC_WRITE 0x1070 0x5 0xFF, etc. to get 80 ns of coincidence window.

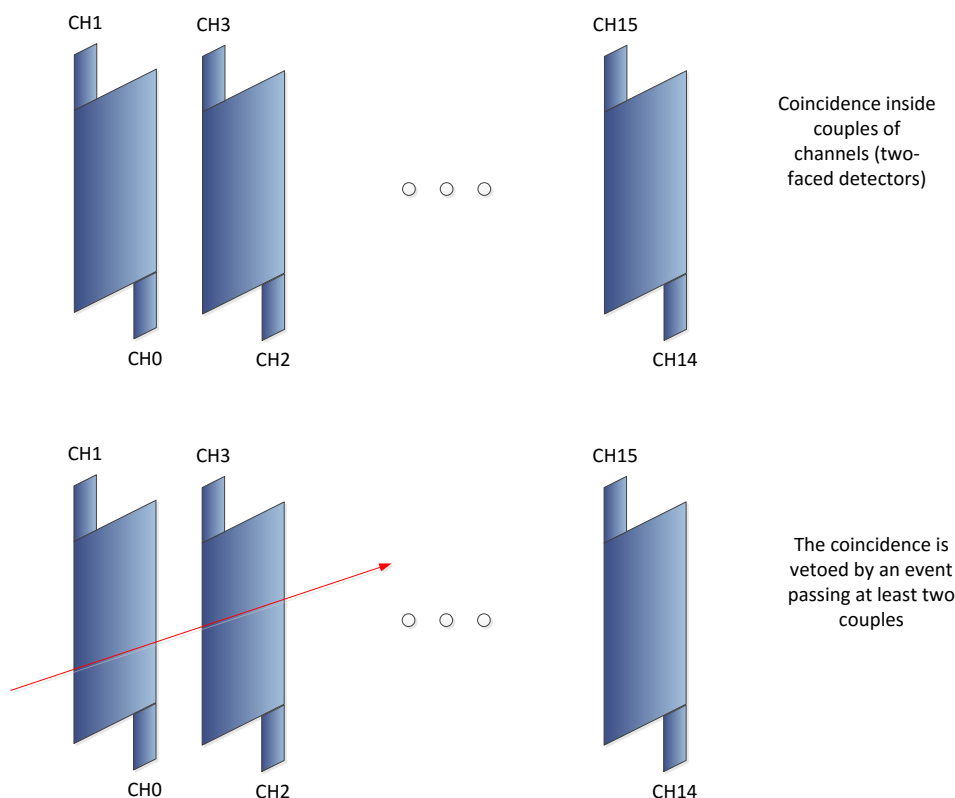


Fig. 5.2: Example of two-sided detectors that acquire in coincidence

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0xC0000 0xC0000
GENERIC_WRITE 0x1180 0xC0000 0xC0000
GENERIC_WRITE 0x1280 0xC0000 0xC0000
GENERIC_WRITE 0x1380 0xC0000 0xC0000
GENERIC_WRITE 0x1480 0xC0000 0xC0000
GENERIC_WRITE 0x1580 0xC0000 0xC0000
GENERIC_WRITE 0x1680 0xC0000 0xC0000
GENERIC_WRITE 0x1780 0xC0000 0xC0000
GENERIC_WRITE 0x1880 0xC0000 0xC0000
GENERIC_WRITE 0x1980 0xC0000 0xC0000
```

```

GENERIC_WRITE 0x1A80 0xC0000 0xC0000
GENERIC_WRITE 0x1B80 0xC0000 0xC0000
GENERIC_WRITE 0x1C80 0xC0000 0xC0000
GENERIC_WRITE 0x1D80 0xC0000 0xC0000
GENERIC_WRITE 0x1E80 0xC0000 0xC0000
GENERIC_WRITE 0x1F80 0xC0000 0xC0000
#
GENERIC_WRITE 0x1070 0xA 0xFF
GENERIC_WRITE 0x1170 0xA 0xFF
GENERIC_WRITE 0x1270 0xA 0xFF
GENERIC_WRITE 0x1370 0xA 0xFF
GENERIC_WRITE 0x1470 0xA 0xFF
GENERIC_WRITE 0x1570 0xA 0xFF
GENERIC_WRITE 0x1670 0xA 0xFF
GENERIC_WRITE 0x1770 0xA 0xFF
GENERIC_WRITE 0x1870 0xA 0xFF
GENERIC_WRITE 0x1970 0xA 0xFF
GENERIC_WRITE 0x1A70 0xA 0xFF
GENERIC_WRITE 0x1B70 0xA 0xFF
GENERIC_WRITE 0x1C70 0xA 0xFF
GENERIC_WRITE 0x1D70 0xA 0xFF
GENERIC_WRITE 0x1E70 0xA 0xFF
GENERIC_WRITE 0x1F70 0xA 0xFF
#
GENERIC_WRITE 0x106C 0x9 0xFF
GENERIC_WRITE 0x116C 0x9 0xFF
GENERIC_WRITE 0x126C 0x9 0xFF
GENERIC_WRITE 0x136C 0x9 0xFF
GENERIC_WRITE 0x146C 0x9 0xFF
GENERIC_WRITE 0x156C 0x9 0xFF
GENERIC_WRITE 0x166C 0x9 0xFF
GENERIC_WRITE 0x176C 0x9 0xFF
GENERIC_WRITE 0x186C 0x9 0xFF
GENERIC_WRITE 0x196C 0x9 0xFF
GENERIC_WRITE 0x1A6C 0x9 0xFF
GENERIC_WRITE 0x1B6C 0x9 0xFF
GENERIC_WRITE 0x1C6C 0x9 0xFF
GENERIC_WRITE 0x1D6C 0x9 0xFF
GENERIC_WRITE 0x1E6C 0x9 0xFF
GENERIC_WRITE 0x1F6C 0x9 0xFF
#
GENERIC_WRITE 0x1084 0x67 0xFF
GENERIC_WRITE 0x1284 0x67 0xFF
GENERIC_WRITE 0x1484 0x67 0xFF
GENERIC_WRITE 0x1684 0x67 0xFF
GENERIC_WRITE 0x1884 0x67 0xFF
GENERIC_WRITE 0x1A84 0x67 0xFF
GENERIC_WRITE 0x1C84 0x67 0xFF
GENERIC_WRITE 0x1E84 0x67 0xFF
#
GENERIC_WRITE 0x8180 0x6FF 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x6FF 0xFFFFFFFF
GENERIC_WRITE 0x8188 0x6FF 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x6FF 0xFFFFFFFF
GENERIC_WRITE 0x8190 0x6FF 0xFFFFFFFF
GENERIC_WRITE 0x8194 0x6FF 0xFFFFFFFF
GENERIC_WRITE 0x8198 0x6FF 0xFFFFFFFF
GENERIC_WRITE 0x819C 0x6FF 0xFFFFFFFF

```

Coincidence inside the VME couples (ch-i & ch-i+1) vetoed by the majority of two or more couples (DPP-PHA)

Here follows the settings for the same example of Sec. **Coincidence inside the VME couples (ch-i & ch-i+1) vetoed by the majority of two or more couples (DPP-PSD)** in case of DPP-PHA firmware. The coincidence window is set equal to 1.6 μ s.



Note: In case of 725 write 0x1084 = 0x64, etc.

```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>False</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1280</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1380</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1480</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1580</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1680</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1780</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
</GenericWrites>
<GenericWrite>
<GWaddr>1880</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1980</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1A80</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1B80</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1C80</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1D80</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1E80</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1F80</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1084</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1184</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1284</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1384</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1484</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1584</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1684</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1784</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1884</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1984</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1A84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1B84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
</xml>
```

```

<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1C84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1D84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1E84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1F84</GWaddr>
<GWdata>C8</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1064</GWaddr>
<GWdata>67</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1264</GWaddr>
<GWdata>67</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1464</GWaddr>
<GWdata>67</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1664</GWaddr>
<GWdata>67</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1864</GWaddr>
<GWdata>67</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1A64</GWaddr>
<GWdata>67</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1C64</GWaddr>
<GWdata>67</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1E64</GWaddr>
<GWdata>67</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>6FF</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8184</GWaddr>
<GWdata>6FF</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>6FF</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8190</GWaddr>
<GWdata>6FF</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8194</GWaddr>
<GWdata>6FF</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8198</GWaddr>
<GWdata>6FF</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>819C</GWaddr>
<GWdata>6FF</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>

```

External Trigger to veto (gate) the acquisition

The acquisition can be synchronized with an external signal, which controls the acquisition according to its logic level high or low. In this case the external signal can be fed into the TRG-IN connector and the TRG_VAL can be set equal to the external trigger itself. Besides the standard use of the TRG-IN as a global trigger, it is possible to program the digitizer to activate the trigger logic for the whole duration of the external trigger. According to the acquisition mode (coincidence/anti-coincidence) the enabled channels can acquire or being vetoed.

An additional register must be set to accomplish this feature (common both to DPP-PSD and DPP-PHA firmware).

Register name	Address	Register bits	Options
Front Panel I/O Control	0x811C	[0]	0: NIM I/O Level 1: TTL I/O Level
		[10]	0: trigger is synchronized with the edge of the TRG-IN signal; 1: trigger is synchronized with the whole duration of the TRG-IN signal. Note: this bit must be used in conjunction with bit[11] = 0.
		[11]	0: TRG-IN signal is processed by the motherboard and sent to mezzanine (default). The trigger logic is then synchronized with TRG-IN; 1: TRG-IN is directly sent to the mezzanines with no mother board processing nor delay. NOTE: if this bit is set to 1, then bit[10] is ignored.

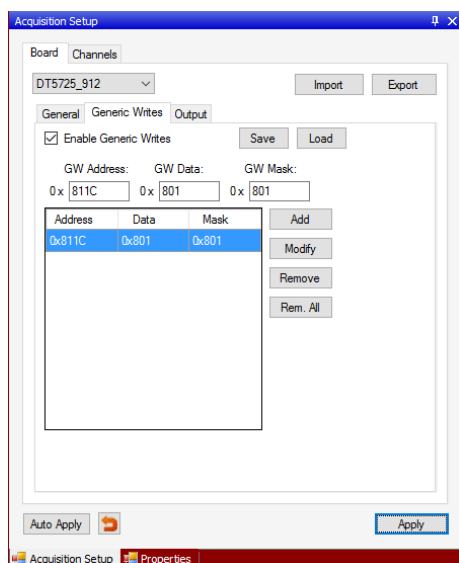
Bit[11] must be set to 1 to ensure that the TRG-IN signal is sent to the channel FPGA (mezzanine) and the acquisition is synchronized with this signal. Bit[0] adjusts the input level choosing between NIM and TTL. For example, in case of DPP-PSD firmware write:

```

GENERIC_WRITE 0x811C 0x800 0x801 # if NIM
GENERIC_WRITE 0x811C 0x801 0x801 # if TTL

```

or, in case of DPP-PHA firmware (TTL level), write:



```

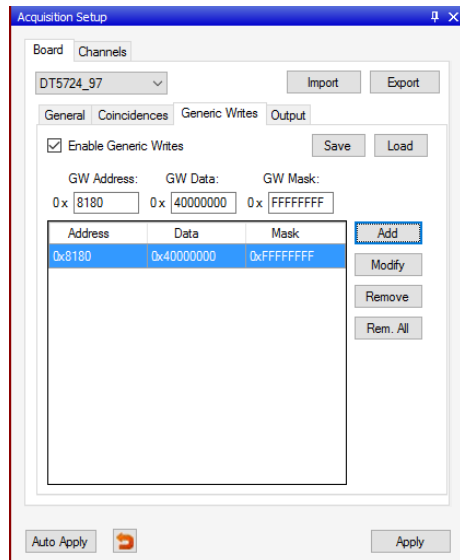
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>811C</GWaddr>
<GWdata>801</GWdata>
<GWmask>801</GWmask>
</GenericWrite>
</GenericWrites>
</config>

```

The TRG_VAL must be set equal to external TRG-IN; the channel will self-trigger on the input pulse, and then it waits for the validation from the external trigger to save the event.
For example, in case of DPP-PSD firmware, write (for channel 0):

```
GENERIC_WRITE 0x8180 0x40000000 0xFFFFFFFF
```

or, in case of DPP-PHA firmware, write:



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

See the examples in the next section for additional details.

Veto from External Trigger - 720 and 751 series with DPP-PSD

Consider an acquisition made by four channels of a DT5720(51) with DPP-PSD firmware, where the acquisition is vetoed by an external TTL signal (on TRG-IN connector). All channels are in anti-coincidence with the external trigger.

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x811C 0x801 0x801
#
GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0xC0000 0xC0000
GENERIC_WRITE 0x1180 0xC0000 0xC0000
GENERIC_WRITE 0x1280 0xC0000 0xC0000
GENERIC_WRITE 0x1380 0xC0000 0xC0000
#
GENERIC_WRITE 0x8180 0x40000000 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x40000000 0xFFFFFFFF
GENERIC_WRITE 0x8188 0x40000000 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x40000000 0xFFFFFFFF
```


Veto from External Trigger - 725 and 730 series with DPP-PSD

Consider an acquisition made by eight channels of a DT5725(30), where the acquisition is vetoed by an external TTL signal (on TRG-IN connector). All channels are in anti-coincidence with the external trigger.

```
# Syntax:
# GENERIC_WRITE 0x<address> 0x<data> 0x<mask>

GENERIC_WRITE 0x811C 0x801 0x801
#
GENERIC_WRITE 0x8000 0x4 0x4
#
GENERIC_WRITE 0x1080 0xC0000 0xC0000
GENERIC_WRITE 0x1180 0xC0000 0xC0000
GENERIC_WRITE 0x1280 0xC0000 0xC0000
GENERIC_WRITE 0x1380 0xC0000 0xC0000
GENERIC_WRITE 0x1480 0xC0000 0xC0000
GENERIC_WRITE 0x1580 0xC0000 0xC0000
GENERIC_WRITE 0x1680 0xC0000 0xC0000
GENERIC_WRITE 0x1780 0xC0000 0xC0000
#
GENERIC_WRITE 0x1084 0x50 0xFF
GENERIC_WRITE 0x1284 0x50 0xFF
GENERIC_WRITE 0x1484 0x50 0xFF
GENERIC_WRITE 0x1684 0x50 0xFF
#
GENERIC_WRITE 0x8180 0x40000000 0xFFFFFFFF
GENERIC_WRITE 0x8184 0x40000000 0xFFFFFFFF
GENERIC_WRITE 0x8188 0x40000000 0xFFFFFFFF
GENERIC_WRITE 0x818C 0x40000000 0xFFFFFFFF
```

Veto from External Trigger - 724, 780 and 781 series with DPP-PHA

An external trigger on the TRG-IN connector is used to veto the acquisition of four channels. Set the "Coincidences" tab as in Fig. 5.3. The coincidence window value has no effect in this case, since the acquisition is vetoed for the whole duration of the external trigger itself.



Note: In case of 780 series select the first two channels only.

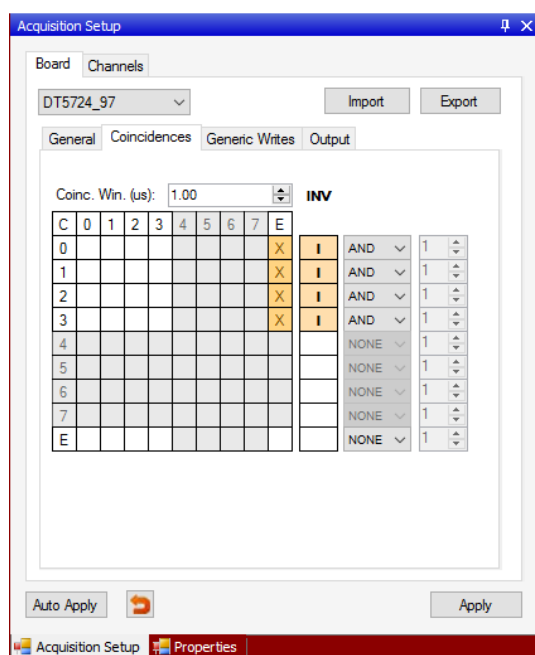
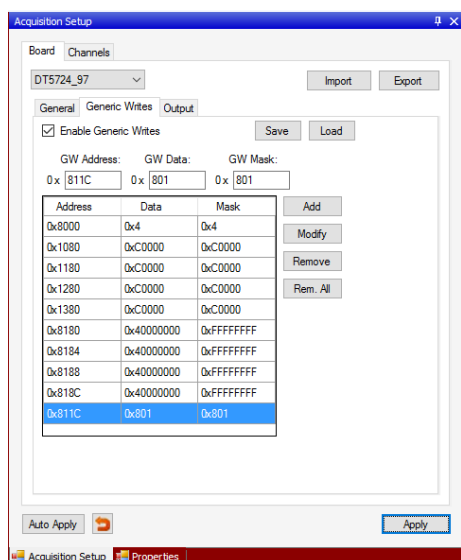


Fig. 5.3: MC² Analyzer configuration of veto from external trigger

Here the list of registers to be enabled.

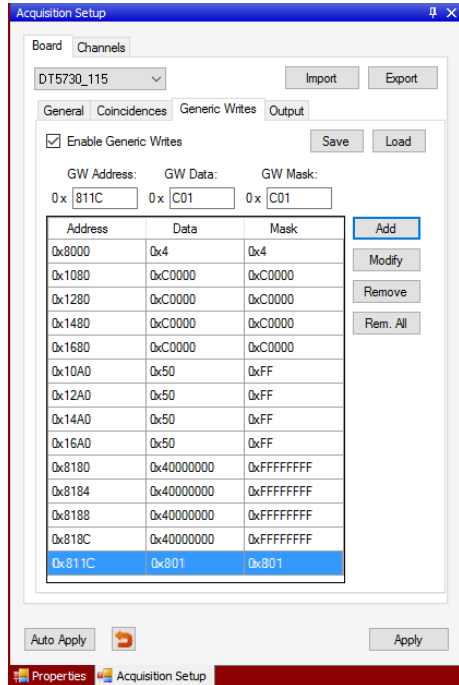


```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1180</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1280</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1380</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8184</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

```
</GenericWrite>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8184</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

Veto from External Trigger - 725 and 730 series with DPP-PHA

An external trigger on the TRG-IN connector is used to veto the acquisition of four channels, in particular ch0, ch2, ch4, and ch6. Set the "Generic Writes" tab as below, or load the settings in the tab itself.



```
<?xml version="1.0"
encoding="UTF-8"?>
<config>
<GenericWrites>
<GWEnable>True</GWEnable>
<GenericWrite>
<GWaddr>8000</GWaddr>
<GWdata>4</GWdata>
<GWmask>4</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1080</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1280</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1480</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>1680</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>10A0</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>12A0</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>14A0</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>16A0</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8184</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

```
<GWaddr>1680</GWaddr>
<GWdata>C0000</GWdata>
<GWmask>C0000</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>10A0</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>12A0</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>14A0</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>16A0</GWaddr>
<GWdata>50</GWdata>
<GWmask>FF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8180</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8184</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

```
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8184</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>8188</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
<GenericWrite>
<GWaddr>818C</GWaddr>
<GWdata>40000000</GWdata>
<GWmask>FFFFFFFF</GWmask>
</GenericWrite>
</GenericWrites>
</config>
```

6 Coincidences with Default Firmware

Data acquisition with Default firmware ensures that whenever an input signal is triggered by one channel, a global trigger from mother board enables all channels to read and save the event. The global trigger can be also configured to make the coincidence logic through the majority operation within a desired time window. This can be done with a firmware register write for most of the digitizer series, with few exceptions. In case of 725 and 730 series it is possible to perform the logic AND/OR of channels of the same couple, as well as a global trigger of couples (majority only). Refer to **[730Manual]** for more details. The 742 series does not have a global trigger generation, therefore it does not have the majority logic either. In case of 743 series it is possible to perform the logic AND/OR between channels of the same couple, and logic AND/OR/MAJORITY among different couples. For more details refer to **[WaveCatcherManual]**. In case of 740 series the coincidence is meant among different groups.

The CAEN software tool *WaveDump*, available at the CAEN web site, can be used to test the majority logic. In particular at the end of the configuration file *WaveDumpConfig.txt* (refer to the specific user manual **[WDMManual]** for more details) it is possible to add the specific register write. Modify the 32 bit "Trigger Source Enable Mask" register (address 0x810C) through the WRITE_REGISTER string, where bits [7:0] enable the specific channels for acquisition, bits [23:20] set the coincidence window width in steps of trigger clock (i.e. 10 ns for 724 series and 8 ns for the others), and bits [26:24] set the majority level. A majority level m enables a global trigger when at least $m + 1$ channels trigger within the coincidence window. Bits [31:30] control the external trigger and the software trigger respectively.



All values MUST be written in hexadecimal.

For example, the coincidence between two channels can be achieved by setting the majority level equal to one. In case you want to have the coincidence of channel 1 and channel 3, with a majority level of 1, and a coincidence window of 10 trigger clocks, you must write in the *WaveDumpConfig.txt* file:

```
WRITE_REGISTER 810C 1A0000A
```

For 761 series the only possible coincidence is between channel 0 and channel 1 of the VME family (refer to **[RD12]**). The register write changes as follows:

```
WRITE_REGISTER 810C 1A00011
```

The following table summarizes the 0x810C bit register description. Refer to 740 and 761 series User Manual for particular cases.

Register name	Address	Register bits	Options
Trigger Source Enable Mask	0x810C	[0]	0 TRG_REQ[0] not propagated 1 TRG_REQ[0] propagated
		[1]	0 TRG_REQ[1] not propagated 1 TRG_REQ[1] propagated
		[2]	0 TRG_REQ[2] not propagated 1 TRG_REQ[2] propagated
		[3]	0 TRG_REQ[3] not propagated 1 TRG_REQ[3] propagated
		[4]	0 TRG_REQ[4] not propagated 1 TRG_REQ[4] propagated
		[5]	0 TRG_REQ[5] not propagated 1 TRG_REQ[5] propagated
		[6]	0 TRG_REQ[6] not propagated 1 TRG_REQ[6] propagated
		[7]	0 TRG_REQ[7] not propagated 1 TRG_REQ[7] propagated
		[19:8]	RESERVED
		[23:20]	Coincidence window
		Majority Level [26:24]	Value for majority
		[28:27]	0 (RESERVED)
		LVDS External Trigger [29]	0 LVDS EXT TRG not propagated 1 LVDS EXT TRG propagated
		External Trigger [30]	0 EXT TRG not propagated 1 EXT TRG propagated
		Software Trigger [31]	0 SOFT TRG not propagated 1 SOFT TRG propagated

7 Technical Support

CAEN makes available the technical support of its specialists at the e-mail addresses below:

support.nuclear@caen.it
(for questions about the hardware)

support.computing@caen.it
(for questions about software and libraries)



CAEN SpA is acknowledged as the only company in the world providing a complete range of High/Low Voltage Power Supply systems and Front-End/Data Acquisition modules which meet IEEE Standards for Nuclear and Particle Physics. Extensive Research and Development capabilities have allowed CAEN SpA to play an important, long term role in this field. Our activities have always been at the forefront of technology, thanks to years of intensive collaborations with the most important Research Centres of the world. Our products appeal to a wide range of customers including engineers, scientists and technical professionals who all trust them to help achieve their goals faster and more effectively.



CAEN S.p.A.
Via Vetràia, 11
55049 Viareggio
Italy
Tel. +39.0584.388.398
Fax +39.0584.388.959
info@caen.it
www.caen.it

CAEN GmbH
Eckehardweg 10
42653 Solingen
Germany
Tel. +49.212.2544077
Mobile +49(0)15116548484
Fax +49.212.2544079
info@caen-de.com
www.caen-de.com

CAEN Technologies, Inc.
1140 Bay Street - Suite 2 C
Staten Island, NY 10305
USA
Tel. +1.718.981.0401
Fax +1.718.556.9185
info@caentechnologies.com
www.caentechnologies.com